

# **WD16 MICROCOMPUTER**

**(Using MCP 3-Chip Microprocessor Set)**

## **PROGRAMMER'S REFERENCE MANUAL**

**WESTERN  DIGITAL**  
**C O R P O R A T I O N**

# **WD1600 MICROCOMPUTER**

**(Using MCP 3-Chip Microprocessor Set)**

**PROGRAMMER'S REFERENCE MANUAL**

**4 OCTOBER 1976**

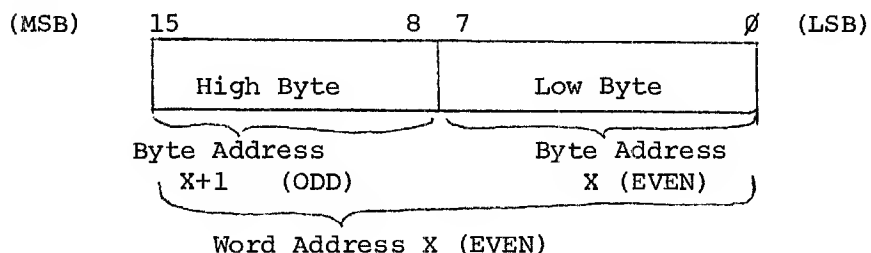
**©1977-WESTERN DIGITAL CORP.  
NEWPORT BEACH, CA. 92663**

## TABLE OF CONTENTS

	<u>PAGE</u>
CHAPTER ONE - GENERAL	1.1
Abbreviations	
Processor Status Word	
Registers	
CHAPTER TWO - INTRODUCTION	2.1
Addressing Modes	
Stack Operations	
Interrupt Lines	
Priority Mask	
External Status Register	
Power Up Options	
Halt Options	
User Bootstrap Routine	
System Error Traps	
Reserved Core Locations	
CHAPTER THREE - OP CODES	3.1
Format 1 Op Codes	
Format 2 Op Codes	
Format 3 Op Codes	
Format 4 Op Codes	
Format 5 Op Codes	
Format 6 Op Codes	
Format 7 Op Codes	
Format 8 Op Codes	
Format 9 Op Codes	
Format 10 Op Codes	
Format 11 Op Codes	
APPENDIX A - Numeric Op Code Table	A1
APPENDIX B - Assembler Notes	B1
APPENDIX C - Programming Notes	C1
APPENDIX D - Microm State Code Functions	D1
APPENDIX E - Op Code Timings	E1

## CHAPTER 1 - GENERAL

The WD1600 microcomputer is a 16 bit machine with both word and byte addressing, an automatic push down hardware stack, vectored interrupt handling, eight 16 bit registers, and PC relative addressing. A byte is defined as 8 bits, and a word is defined as 2 bytes. A memory address increment of one is an increment of 1 byte. An address increment of two is an increment of 1 word. Word addresses always start on even bytes. For any memory location the even byte is the least significant byte. Bit 0 is defined as the LSB of a memory location.



Unless otherwise stated, word addressing is implied. All addresses and op codes are done in hex unless otherwise stated. All hex numbers are enclosed within double quotes.

### LEGEND OF ABBREVIATIONS

REG = Register

SRC = Source Address

(SRC) = Contents of Source Address

DST = Destination Address

(DST) = Contents of Destination Address

(SRC)<sub>B</sub> = Contents of Source Byte Address

(DST)<sub>B</sub> = Contents of Destination Byte Address

$\overline{x}$  = Ones Complement of X  
 $\neg x$  = Twos Complement of X  
 $\Delta$  = Logical And  
 $\nabla$  = Logical Or  
 $\nabla$  = Exclusive or  
 $@$  = Indirect  
 $\downarrow$  = Push  
 $\uparrow$  = Pop  
 $\leftarrow$  = Destination Direction  
 $+$  = Addition  
 $-$  = Subtraction  
 $*$  = Multiplication  
 $/$  = Division  
 $:$  = Double Precision Chain Link

#### PROCESSOR STATUS WORD

A 16 bit Processor Status (PS) Word exists. The format is as follows:

15	8	7	4	3	2	1	0
Ext. Status Reg.	ALU			N	Z	V	C

Where bits 8-15 are the contents of the external status register (see chapter 2), bits 4-7 are the status of the microprocessor ALU flags, and bits 0-3 are the status of the condition indicators at the time the PS is formed. The ALU flags are of no use or concern to the programmer. They are stored along with the condition indicators automatically as a function of the micro-op. The four condition flags are updated during the execution of most op codes, and are used by the branch instructions to test for valid branch conditions. The exact status of each indicator is defined along with the descriptions of individual op codes in chapter 3. In general, however, the indicators are set by the following conditions:

N = set if the MSB of the result is set.  
 Z = set if the result is zero.  
 V = set if arithmetic overflow (underflow) occurs during addition (subtraction).  
     Set to exclusive-or of N and C indicators otherwise.  
 C = set if carry (borrow) occurs during addition (subtraction). Also set  
     to last bit shifted out during a shift operation.

## REGISTERS

There are 8 registers in the WD1600. All are 16 bits long. Six can be used as either accumulators or index registers, one is the stack pointer (SP), and one is the program counter (PC). The registers are numbered R0 - R7 with R6 = SP and R7 = PC. The register set is usually referred to in the following manner: R0 - R5, SP, PC.



## CHAPTER TWO - INTRODUCTION

### ADDRESSING MODES

In general there are 8 addressing modes for both source and destination addressing. Not all op codes accept all 8 modes (see chapter 3). Those that do use the following format: 3 bits for the index register (R0 - R5, SP, PC) and 3 bits for the mode. The mode bits are the upper 3 bits of the 6 bit set. The modes are defined below. The numbers in parenthesis refer to notes that follow the definitions.

MODE	NAME	SYMBOLIC	DESCRIPTION
0	Direct Register	REG	REG is or contains operand.
1	Indirect Register	@REG	REG contains address of operand.
2	Auto-increment	(REG)+	REG contains address of operand. REG is post-incremented (1).
3	Auto-increment deferred	@(REG)+	REG contains address of address of operand. REG is post-incremented by 2.
4	Auto-decrement	-(REG)	REG is predecremented (1). REG then contains address of operand.
5	Auto-decrement deferred	@-(REG)	REG is predecremented by 2. REG then contains address of address of operand.
6	Indexed register	X(REG)	Contents of REG plus X is address of operand (2).
7	Indexed register deferred	@X(REG)	Contents of REG plus X is address of address of operand (2).

NOTE 1: For word operations the increment/decrement is 2. For byte operations the increment/decrement is 1 unless the index register is SP or PC. In this case the increment/decrement is always 2.

NOTE 2: The contents of REG remain unchanged.

When using PC as the index register the assembler accepts the following 4 formats in place of the formats mentioned above for ease of programming.

MODE	NAME	SYMBOLIC	DESCRIPTION
2	Immediate	#N	Operand N follows op code.
3	Absolute	@#N	Address of operand is N and it follows the op code in memory.
6	Relative	A	PC relative offset to address A, which contains operand, follows op code.
7	Relative deferred @A		PC relative offset to address A, which contains address of operand, follows the op code.

The 8 modes are referred to as Source Mode 0 to Source Mode 7 (SM0 -SM7) and Destination Mode 0 to Destination Mode 7 (DM0 -DM7). In Chapter 3 these modes are referred to in general terms during op code definitions as "SRC" and "DST".



## STACK OPERATIONS

Although automatic stack operations are provided for, no specific area of memory is set aside for the stack. The user must assign an area of memory by loading the stack pointer with the top address of the designated stack area. Stack operations are push-down pop-up operations with predecrements and post-increments of SP. Stack operations may also be executed explicitly by using SP as an index register with op codes that allow SM0 - SM7 and/or DM0 - DM7 addressing.

When pushing the PS the word is formed just prior to the push. When popping the PS the condition indicators and interrupt enable flag are set to the status of the appropriate bits in the popped PS. Other than that the popped PS goes nowhere. Unless otherwise stated popping the PS from the stack performs the above mentioned operations and only the above mentioned operations.

When pushing the PC onto the stack PC will be set to the address of the op code that follows the op code that caused the push. There are cases where some op code formats can alter this rule. They generally involve advanced programming techniques. A few are mentioned in appendix C. In particular, system errors that are caused by programming errors and not real time error conditions will push a PC that points to the op code that follows the op code that caused the error. The stored PC must be decremented by two to get the address of the offending op code.

## INTERRUPT LINES

There are 4 interrupt lines available to the system. They are labeled I0 - I3. These lines are assigned functions as follows:

I0 = Vectored interrupt line  
I1 = Nonvectored interrupt line  
I2 = Enable/disable for I0 and I1.  
I3 = Halt switch

The priority among the lines is as follows:

I3, I1ΔI2, I0ΔI2.

Note that I3 is always enabled. Note also that the nonvectored interrupt has priority over the vectored interrupt. The system is currently set up so that power fail and a real time clock can be assigned to I1, and up to 16 devices assigned to I0.\* The two interrupts operate as follows:

### A) Nonvectored Interrupt (I1)

PS and PC are pushed onto the stack. I2 is disabled. The external status register is tested for a power fail. If power fail is true PC is fetched from location "14". If power fail is false PC is fetched from location "2A", and a microm state code is transmitted to clear the line clock (see appendix D).

### B) Vectored Interrupt (I0)

PS and PC are pushed onto the stack. I2 is disabled. An Interrupt Acknowledge is executed, and the device code of the interrupting device is read in and stripped to bits 1-4. PC is fetched from location

\*NOTE: Although only a 4bit device code is currently used, a minor microm change can allow a device code of from 1-15 bits.

"28" and the device code is added to it. The contents of this intermediate location are read in and added to PC to form the final address. Each intermediate location is a table entry that contains the PC relative offset from the start of the device handler routine to itself. The absolute address of the start of the table is in location "28".

#### PRIORITY MASK

Associated with the interrupts is a priority interrupt mask. This is a 16 bit mask where each bit position represents a priority level. Each priority level can be assigned to one or more devices. A one in any bit position can represent an interrupt enable or disable for its associated devices as the hardware dictates. The SAVS, RSTS, and MSKO op codes each alter the mask. When the mask is altered it is written into location "2E" for storage. While the mask is on the bus a microm state code is transmitted (see appendix D) to signal the I/O devices that a new mask is being transmitted. Each device can then look at its assigned mask bit while the memory write to location "2E" is taking place. Whether or not the mask feature is actually used by the I/O devices in no way alters the operations of the op codes mentioned above.

#### EXTERNAL STATUS REGISTER

As a part of the hardware external to the CPU the External Status Register supplies the CPU, upon demand, with information about the status of certain hardware areas. This register is gated onto the bus when its associated microm state code is present (see appendix D). The format of the register is as follows:

- Bit 7 = Power Fail Status
- Bit 6 = Bus Error (Time Out) Status
- Bit 5 = Parity Error Status
- Bit 4 = I2 Interrupt Line Status
- Bit 3 = Halt Option Jumper #2
- Bit 2 = Halt Option Jumper #1
- Bit 1 = Power Up Option Jumper #2
- Bit 0 = Power Up Option Jumper #1

Bits 8-15 are don't care. Bits 5-7 are real time error conditions that also generate a system reset (see next section). Bit 4 is the interrupt enable status. The jumpers can be logic units, switches, or hard wired jumpers as the user wishes. The various options associated with the 4 jumpers are discussed later.

#### POWER UP OPTIONS

A system reset indicate one of 4 conditions: power fail, bus error, parity error, or power up. There are 2 levels of power fail possible in this system (see appendix C): minor and major. Only a major power fail generates a system reset. Both types set bit 7 in the External Status Register. The following steps are performed after a system reset.

- A1) Trace and wait flags are reset if on.
- A2) The external Status Register is fetched.

- A3) The Line-clock-clear state code is transmitted.
- A4) I2 is reset.
- A5) If power fail bit is set go to D1.
- A6) If bus error bit is set go to C1.
- A7) If parity error bit is set go to B1.
- A8) Go to D2 otherwise.

- B1) Push PS and PC onto stack.
- B2) Fetch PC from location "12" and begin execution.
- C1) Push PS and PC onto stack.
- C2) Fetch PC from location "18" and begin execution.
- D1) Wait until power fail status = 0.
- D2) Send a system reset microm state code.
- D3) Wait 300 cycles.
- D4) Execute power up option 1,2,3 or 4 per jumpers.

For a proper initial power up either bit 7 must be set or bits 5-7 must be reset when the system reset line is released.

The 4 power up options are as follows:

JUMPERS	OPERATION
00	Execute user bootstrap routine.
01	Pick up R0-R5, SP, PC, and PS from memory locations 0-"10".
10	Execute selected halt option.
11	Fetch PC from location "16".

#### HALT OPTIONS

When the halt switch (I3) is set during program execution one of 4 halt options is selected. The halt op code\* and power up option #2 also select the halt option specified. The options are as follows:

JUMPERS	OPERATION
00	Execute user bootstrap routine.
01	Save R0-R5, SP, PC and PS in memory locations 0-"10". Wait until I3 = 0, then restore R0-R5, SP, PC and PS from memory locations 0-"10".
10	Lock up processor (requires a system reset to clear).
11	Fetch new PC from location "16".

\*NOTE: Conditional. See Chapter 3.

#### USER BOOTSTRAP ROUTINE

When the user bootstrap routine is selected as an option the system creates the starting address by placing address "C000" in PC and then replacing bits 8-13 with the contents of the 6 bit External Address Register. This register is gated in with a microm status code (see appendix D).

It allows the user 64 different starting addresses in the range "C000" to "FF00".

### SYSTEM ERROR TRAPS

With the exception of the major power fail error that is a function of a system reset, all error conditions perform a common routine as outlined below. A non-vectored interrupt and some op codes also use this routine. The numbers in parenthesis refer to notes that follow the table.

- 1) PS is pushed onto the stack
- 2) PC is pushed onto the stack
- 3) PC is fetched from location X where "X" is from the following table

- |             |      |   |
|-------------|------|---|
| (1) (2) (3) | "12" | for bus error PC                        |
| (1) (2) (3) | "14" | for nonvectored interrupt power fail PC |
| (1) (2) (3) | "18" | for parity error PC                     |
| (1) (2) (3) | "1A" | for reserved op code error PC           |
| (1) (2) (3) | "1C" | for illegal op code format error PC     |
| (1) (2) (3) | "1E" | for XCT error PC                        |
| (1) (2)     | "20" | for XCT trace PC                        |
| (1) (2) (3) | "2A" | for nonvectored interrupt PC            |
| (1) (2)     | "2C" | for BPT PC                              |

NOTE 1: wait flag reset if on

NOTE 2: trace flag reset if on

NOTE 3: interrupt enable (I2) reset if on

The meaning of the wait and trace flags is discussed in chapter 3. Note that the nonvectored interrupt power fail PC is a minor power fail condition, not a major one. See appendix C for full detail on how to include both major and minor power fail conditions in the hardware.

### RESERVED CORE LOCATIONS

The following is a complete list of memory locations that are reserved for specific system functions or options. Byte addresses are given.

<u>LOCATIONS</u>	<u>RESERVED FUNCTION</u>
0 - "11"	R0 - R5, SP, PC and PS for power up/halt options
"12" - "13"	bus error PC
"14" - "15"	nonvectored interrupt power fail PC
"16" - "17"	power up/halt option power restore PC
"18" - "19"	parity error PC
"1A" - "1B"	reserved op code PC
"1C" - "1D"	illegal op code format PC
"1E" - "1F"	XCT error PC
"20" - "21"	XCT trace PC
"22" - "23"	SVCA table address
"24" - "25"	SVCB PC
"26" - "27"	SVCC PC
"28" - "29"	vectored interrupt (I0) table address
"2A" - "2B"	nonvectored interrupt (I1) PC
"2C" - "2D"	BPT PC
"2E" - "2F"	I/O priority interrupt mask
"30" - "3F"	reserved for floating point option

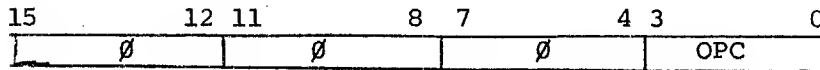


### CHAPTER 3 - OP CODES

This chapter is divided into a number of sections, each representing one class of op codes. At the beginning of each section there is a detailed description of the format for that class. A list of op codes and their base numeric values, less arguments, is also included. A detailed description of each op code in the class then follows.

#### FORMAT 1 OP CODES

Single word - no arguments



There are 16 op codes in this class representing op codes "0000" to "000F". Each is a one word op code with no arguments with the exception of the SAVS op code which is a two word op code. Word two of the SAVS op code is the I/O priority interrupt mask. The op codes and their mnemonics are:

BASE OP CODE	MNEMONIC
0000	NOP
0001	RESET
0002	IEN
0003	IDS
0004	HALT
0005	XCT
0006	BPT
0007	WFI
0008	RSVC
0009	RRTT
000A	SAVE
000B	SAVS
000C	REST
000D	RRTN
000E	RSTS
000F	RTT
NOP	NO OPERATION
FORMAT:	NOP
FUNCTION:	No operations are performed
INDICATORS:	Unchanged
RESET	I/O RESET
FORMAT:	RESET
FUNCTION:	An I/O reset pulse is transmitted
INDICATORS:	Unchanged

IEN	INTERRUPT ENABLE
FORMAT:	IEN
FUNCTION:	The interrupt enable (I2) flag is set. Allows one more instruction to execute before interrupts are recognized.
INDICATORS:	Unchanged
IDS	INTERRUPT DISABLE
FORMAT:	IDS
FUNCTION:	The interrupt enable (I2) flag is reset. This instruction can honor interrupts, but the I2 bit in the PS that is stored on the stack is reset if an interrupt occurs.*
INDICATORS:	Unchanged
*NOTE: On some machines I2 will be set or reset during the IEN or IDS . If so the change will be valid immediately, not one op code later.	
HALT	HALT
FORMAT:	HALT
FUNCTION:	Tests the status of the Power Fail bit in the external status register. If the bit is set it is assumed that the HALT occurred in a power fail routine, and the following operations occur: 1) The interrupt enable (I2) flag is reset 2) The CPU waits until the Power Fail bit is reset 3) PC is fetched from location "16", and program execution begins at this new location If the power fail bit is reset then the CPU waits until the halt switch (I3) is set. At that time the selected halt option (see chapter 2) is executed. The interrupt enable flag is also reset.
INDICATORS:	Unchanged
XCT	EXECUTE SINGLE INSTRUCTION
FORMAT:	XCT
OPERATION:	PC ← @SP, SP ↑ PS ← @SP, SP ↑ Trace flag set, execute op code ↓SP, @SP ← PS ↓SP, @SP ← PC Trace flag reset PC ← (loc "20") if no error PC ← (loc "1E") if error
FUNCTION:	PC and PS are popped from the stack, but I2 is not altered. The trace flag, which disables all interrupts except I3, is set. The op code is executed. PS and PC are pushed back onto the stack, and PC is fetched from location "20". The trace flag is reset. If the program tries to execute a HALT, XCT, BPT, or WFI the attempt is aborted, PS and PC are

pushed onto the stack, and PC is fetched from location "1E" instead.  
I2 is also reset.

INDICATORS: Depends upon executed op code

---

**BPT** **BREAKPOINT TRAP**

---

FORMAT: BPT  
 OPERATION:  $\downarrow SP, @SP \leftarrow PS$   
 $\downarrow SP, @SP \leftarrow PC$   
 $PC \leftarrow (loc\ "2C")$   
 FUNCTION: PS and PC are pushed onto the stack. PC is  
 fetched from location "2C"  
 INDICATORS: Unchanged

---

**WFI** **WAIT FOR INTERRUPT**

---

FORMAT: WFI  
 FUNCTION: The CPU loops internally without accessing  
 the data bus until an interrupt occurs. Program  
 execution continues with the op code that follows  
 the WFI after the interrupt has been serviced.  
 The interrupt enable flag is also set.  
 INDICATORS: Unchanged

---

**SAVE** **SAVE REGISTERS**

---

FORMAT: SAVE  
 OPERATION:  $\downarrow SP, @SP \leftarrow R5$   
 $\downarrow SP, @SP \leftarrow R4$   
 $\downarrow SP, @SP \leftarrow R3$   
 $\downarrow SP, @SP \leftarrow R2$   
 $\downarrow SP, @SP \leftarrow R1$   
 $\downarrow SP, @SP \leftarrow R0$   
 FUNCTION: Registers R5 to R0 are pushed onto the stack.  
 INDICATORS: Unchanged.

---

**SAVS** **SAVE STATUS**

---

FORMAT: SAVS MASK  
 OPERATION: SAVE  
 $\downarrow SP, @SP \leftarrow (loc\ "2E")$   
 $(loc\ "2E") \leftarrow (loc\ "2E") \vee mask$   
 MSKO  
 IEN  
 FORMAT: Registers R5 to R0 and the priority mask in location  
 "2E" are pushed onto the stack. The old and new masks  
 are ORED together and placed in location "2E".  
 A mask out state code (see appendix D) is transmitted  
 and the interrupt enable (I2) flag is set.  
 INDICATORS: Unchanged

---

**REST** **RESTORE REGISTERS**

---

FORMAT: REST  
 OPERATION:  $R0 \leftarrow @SP, SP \uparrow$   
 $R1 \leftarrow @SP, SP \uparrow$   
 $R2 \leftarrow @SP, SP \uparrow$



FUNCTION: Registers R0 to R5 are popped from the stack,  
INDICATORS: Unchanged

FORMAT:	RTT
OPERATION:	PC ← @SP, SP ↑ PS ← @SP, SP ↑
FUNCTION:	PC and PS are popped from stack
INDICATORS:	N = Set per PS bit 3 Z = Set per PS bit 2 V = Set per PS bit 1 C = Set per PS bit 0

FORMAT:	RRTN
OPERATION:	REST
FUNCTION:	PC ← @SP, SP↑ Registers R0 to R5 and PC are popped from the stack
INDICATORS:	Unchanged

```

FORMAT:                RRTT
OPERATION:              REST
                        RTT
FUNCTION:               Registers R0 to R5, PC and PS are popped
                        from the stack.
INDICATORS:             Set per PS bits 0 - 3

```

FORMAT:	RSTS
OPERATION:	(LOC "2E") ← @SP, SP ↑
	MSKO
	REST
	RTT
FUNCTION:	The priority mask is popped from the stack and restored to location "2E". A MASK OUT state code (See Appendix D) is transmitted. Registers R0 to R5, PC and PS are popped from the stack.
INDICATORS:	Set per PS bits 0 - 3

```

FORMAT:                                RSVC
OPERATION:                             REST
                                         SP↑
                                         RTT

```

FUNCTION:

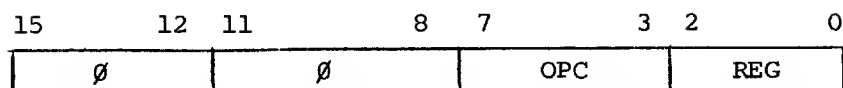
Registers R0 to R5, PC and PS are popped from the stack with the saved SP bypassed.

INDICATORS:

Set per PS bits 0 - 3

## FORMAT 2 OP CODES

### SINGLE WORD - 3 BIT REGISTER ARGUMENT



There are 4 op codes in this class representing op codes "0010" to "002F". Each is a one word op code with a single 3 - bit register argument. The op codes and their mnemonics are:

BASE OP CODE	MNEMONIC
0010	IAK
0018	RTN
0020	MSKO
0028	PRTN

IAK	INTERRUPT ACKNOWLEDGE
-----	-----------------------

FORMAT:	IAK            REG
FUNCTION:	An interrupt acknowledge (READ and IACK) is executed, and the 16 bit code that is returned is placed in REG unmodified. Used with the nonvectored interrupt when the user does not wish to use the vectored format.
INDICATORS:	Unchanged

RTN	RETURN FROM SUBROUTINE
-----	------------------------

FORMAT:	RTN            REG
OPERATION:	PC    ←    REG REG   ←    @SP, SP ↑
FUNCTION:	The linkage register is placed in PC and the saved linkage register is popped from the stack. The register used must be the same one that was used for the subroutine call.
INDICATORS:	Unchanged

MSKO	MASK OUT
------	----------

FORMAT:	MSKO            REG
OPERATION:	(LOC "2E" ) ← REG MSKO
FUNCTION:	The contents of REG are written into location "2E" and a MASK OUT state code (see appendix D) is transmitted.
INDICATORS:	Unchanged

PRTN	POP STACK AND RETURN
------	----------------------

FORMAT:	PRTN            REG
OPERATION:	TMP ← @SP SP ← SP+(TMP*2) RTN    REG

FUNCTION:

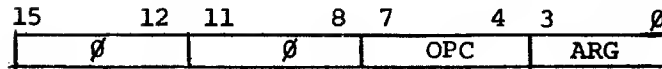
Twice the value of the top word on the stack is added to SP, and a standard RTN call is then executed.

INDICATORS:

Unchanged

### FORMAT 3 OP CODES

#### SINGLE WORD - 4 BIT NUMERIC ARGUMENT



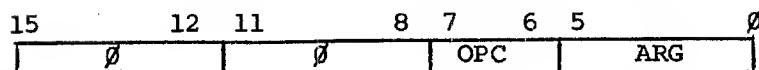
There is only one op code in this class representing op codes "0030" to "003F". It is a one word op code with a 4-bit numeric argument.

BASE OP CODE	MNEMONIC
0030	LCC
LCC	LOAD CONDITION CODES

FORMAT:	LCC    ARG
FUNCTION:	The 4 indicators are loaded from bits 0-3 of the op code as specified.
INDICATORS:	N = Set per bit 3 of op code Z = Set per bit 2 of op code V = Set per bit 1 of op code C = Set per bit 0 of op code

# FORMAT 4 OP CODES

## SINGLE WORD - 6 BIT NUMERIC ARGUMENT



There are 3 op codes in this class representing op codes "0040" to "00FF". All 3 are supervisor calls. All 3 are one word op codes with a 6-bit numeric argument.

BASE OP CODE	MNEMONIC
0040	SVCA
0080	SVCB
00C0	SVCC
SVCA	SUPERVISOR CALL "A"

**FORMAT:** SVCA ARG  
**OPERATION:**  $\downarrow SP, @SP \leftarrow PS; \downarrow SP, @SP \leftarrow PC$   
 $PC \leftarrow (LOC "22") + (ARG * 2)$   
 $PC \leftarrow PC + @PC$   
**FUNCTION:** PS and PC are pushed onto the stack. The contents of location "22" plus twice the value of the argument (which is always positive) is placed in PC to get the table address. The contents of the table address is added to PC to get the final destination address. Each table entry is the relative offset from the start of the desired routine to itself.  
**INDICATORS:** Unchanged

SVCB	SUPERVISOR CALL "B"
SVCC	SUPERVISOR CALL "C"

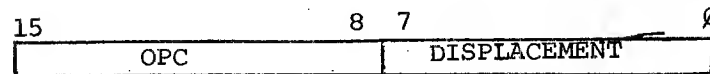
**FORMAT:** SVCB ARG  
 SVCC ARG  
**OPERATION:**  $TMPA \leftarrow SP$   
 $\downarrow SP, @SP \leftarrow PS$   
 $\downarrow SP, @SP \leftarrow PC$   
 $TMPB \leftarrow SP$   
 $\downarrow SP, @SP \leftarrow TMPA$   
 SAVE  
 $R1 \leftarrow TMPB$   
 $R5 \leftarrow ARG * 2$   
 $PC \leftarrow (LOC "24")$  if SVCB  
 $PC \leftarrow (LOC "26")$  if SVCC  
**FUNCTION:** PS and PC are pushed onto the stack. The value of SP at the start of op code execution is the pushed followed by registers R5 to R0. The address of the saved PC is placed in R1, and twice the value of the 6-bit positive argument is placed in R5.

INDICATORS:

PC is loaded from location "24"  
for SVCB or "26" for SVCC.  
Unchanged.

## FORMAT 5 OP CODES

SINGLE WORD - 8 BIT SIGNED NUMERIC ARGUMENT



There are 15 op codes in this class representing op codes "0100" to "07FF" and "8000" to "87FF". All are branches with a signed 8 bit displacement that represents the word offset from PC (which points to the op code that follows) to the desired branch location. The op codes consist on one unconditional branch, 8 signed conditional branches, and 6 unsigned conditional branches. No op code in this class modifies any of the indicator flags. Maximum branch range is +128, -127 words from the branch op code.

BASE OP CODE	MNEMONIC
0100	BR
0200	BNE
0300	BEQ
0400	BGE
0500	BLT
0600	BGT
0700	BLE
8000	BPL
8100	BMI
8200	BHI
8300	BLOS
8400	BVC
8500	BVS
8600	BCC, BHIS
8700	BCS, BLO

### BR BRANCH UNCONDITIONALLY

FORMAT:	BR DEST
OPERATION:	PC ← PC + (DISP *2)
FUNCTION:	Twice the value of the signed displacement is added to PC.

### SIGNED BRANCHES

#### BNE BRANCH IF NOT EQUAL TO ZERO

FORMAT:	BNE DEST
OPERATION:	IF Z = 0, PC ← PC + (DISP *2)

#### BEQ BRANCH IF EQUAL TO ZERO

FORMAT:	BEQ DEST
OPERATION:	IF Z = 1, PC ← PC + (DISP *2)

#### BGE BRANCH IF GREATER THAN OR EQUAL TO ZERO

FORMAT:	BGE DEST
OPERATION:	IF N∇V = 0, PC ← PC + (DISP *2)



<u>BLT</u>	<u>BRANCH IF LESS THAN ZERO</u>
FORMAT:	BLT    DEST
OPERATION:	IF $N\bar{V}V = 1$ , $PC \leftarrow PC + (DISP * 2)$
<u>BGT</u>	<u>BRANCH IF GREATER THAN ZERO</u>
FORMAT:	BGT    DEST
OPERATION:	IF $Z \vee (N\bar{V}V) = 0$ , $PC \leftarrow PC + (DISP * 2)$
<u>BLE</u>	<u>BRANCH IF LESS THAN OR EQUAL TO ZERO</u>
FORMAT:	BLE    DEST
OPERATION:	IF $Z \vee (N\bar{V}V) = 1$ , $PC \leftarrow PC + (DISP * 2)$
<u>BPL</u>	<u>BRANCH IF PLUS</u>
FORMAT:	BPL    DEST
OPERATION:	IF $N = 0$ , $PC \leftarrow PC + (DISP * 2)$
<u>BMI</u>	<u>BRANCH IF MINUS</u>
FORMAT:	BMI    DEST
OPERATION:	IF $N = 1$ , $PC \leftarrow PC + (DISP * 2)$
<u>UNSIGNED BRANCHES</u>	
<u>BHI</u>	<u>BRANCH IF HIGHER</u>
FORMAT:	BHI    DEST
OPERATION:	IF $CVZ = 0$ , $PC \leftarrow PC + (DISP * 2)$
<u>BLOS</u>	<u>BRANCH IF LOWER OR SAME</u>
FORMAT:	BLOS    DEST
OPERATION:	IF $CVZ = 1$ , $PC \leftarrow PC + (DISP * 2)$
<u>BVC</u>	<u>BRANCH IF OVERFLOW CLEAR</u>
FORMAT:	BVC    DEST
OPERATION:	IF $V = 0$ , $PC \leftarrow PC + (DISP * 2)$
<u>BVS</u>	<u>BRANCH IF OVERFLOW SET</u>
FORMAT:	BVS    DEST
OPERATION:	IF $V = 1$ , $PC \leftarrow PC + (DISP * 2)$
<u>BCC</u>	<u>BRANCH IF CARRY CLEAR</u>
<u>BHIS</u>	<u>BRANCH IF HIGHER OR SAME</u>
FORMAT:	BCC    DEST
OPERATION:	BHIS    DEST IF $C = 0$ , $PC \leftarrow PC + (DISP * 2)$

BCS	BRANCH IF CARRY SET
BLO	BRANCH IF LOWER

FORMAT:	BCS	DEST
	BLO	DEST

OPERATION:	IF C = 1, PC $\leftarrow$ PC + (DISP *2)
------------	--

# FORMAT 6 OP CODES

SINGLE WORD - SINGLE OPS - SPLIT FIELD - DMØ ONLY

15	9	8	6	5	4	3	0
OPC BASE		REG		OPC		COUNT	

There are 12 op codes in this class representing op codes "Ø8ØØ" to "Ø9FF", "88ØØ" to "89FF", and "8EØØ" to "8FFF". There are 4 immediate mode op codes with a register as a destination, 4 multiple count single register shifts, and 4 multiple count double register shifts. In all op codes the actual count (or number in the case of the immediates) is the value of bits Ø - 3 plus one. Count is always a positive number in the range 1 - "1Ø", but it is stored in the op code as Ø - "F". All of these op codes are one word op codes with the op codes themselves split between bits 9-15 and 4-5.

In the case of the double shifts the 32 bit number (REG+1) : (REG) is the operand. If REG = PC then (REG+1) = RØ.

BASE OP CODE	MNEMONIC
Ø8ØØ	ADDI
Ø81Ø	SUBI
Ø82Ø	BICI
Ø83Ø	MOVI
88ØØ	SSRR
881Ø	SSLR
882Ø	SSRA
883Ø	SSLA
8EØØ	SDRR
8E1Ø	SDLR
8E2Ø	SDRA
8E3Ø	SDLA
ADDI	ADD IMMEDIATE
FORMAT:	ADDI    NUMBER, REG
OPERATION:	REG ← REG + COUNT + 1
FUNCTION:	The stored number plus one is added to the destination register.
INDICATORS:	N = Set if bit 15 of the result is set Z = Set if the result = Ø V = Set if arithmetic overflow occurs; i.e. set if both operands were positive and the sign of the result is negative C = Set if a carry was generated from bit 15 of the result
SUBI	SUBTRACT IMMEDIATE
FORMAT:	SUBI    NUMBER, REG
OPERATION:	REG ← REG - (COUNT +1)
FUNCTION:	The stored number plus one is subtracted from the destination register.

INDICATORS:           N = Set if bit 15 of the result is set  
                       Z = Set if the result = 0  
                       V = Set if arithmetic underflow occurs; i.e. set  
                           if the operands were of opposite signs and  
                           the sign of the result is positive  
                       C = Set if a borrow was generate from bit 15  
                           of the result

---

BICI                                   BIT CLEAR IMMEDIATE

---

FORMAT:               BICI       NUMBER, REG  
 OPERATION:           REG ← REG Δ(COUNT + 1)  
 FUNCTION:            The stored number plus one is one's complemented  
                       and ANDED to the destination register  
 INDICATORS:          N = Set if bit 15 of the result is set  
                       Z = Set if the result = 0  
                       V = Reset  
                       C = Unchanged

---

MOVI                                  MOVE IMMEDIATE

---

FORMAT:               MOVI       NUMBER, REG  
 OPERATION:           REG ← COUNT + 1  
 FUNCTION:            The stored number plus one is placed in  
                       the destination register  
 INDICATORS:          N = Reset  
                       Z = Reset  
                       V = Reset  
                       C = Unchanged

---

SSRR                                 SHIFT SINGLE RIGHT ROTATE

---

FORMAT:               SSRR       REG, COUNT  
 FUNCTION:            A 17-bit right rotate is done stored count+1  
                       times on REG:C-Flag. The C-Flag is shifted into  
                       bit 15 of REG, and the C-Flag gets the last bit  
                       shifted out of REG bit 0.  
 INDICATORS:          N = Set if bit 7 of REG is set  
                       Z = Set if REG = 0  
                       V = Set to exclusive or of N and C flags  
                       C = Set to the value of the last bit shifted  
                       out of REG bit 0

---

SSLR                                 SHIFT SINGLE LEFT ROUTINE

---

FORMAT:               SSLR       REG, COUNT  
 FUNCTION:            A 17-bit left rotate is done stored count+1  
                       times on C-Flag:REG. The C-Flag is shifted  
                       into bit 0 of REG and the C-Flag gets the  
                       last bit shifted out of REG bit 15.  
 INDICATORS:          N = Set if bit 15 of REG is set  
                       Z = Set if REG = 0  
                       V = Set to exclusive or of N and C flags  
                       C = Set to the value of the last bit shifted  
                       out of REG bit 15.

---

SSRA                      SHIFT SINGLE RIGHT ARITHMETIC

---

FORMAT:                      SSRA     REG, COUNT  
FUNCTION:                    A 17-bit right arithmetic shift is done stored count+1 times on REG:C-Flag. Bit 15 of REG is replicated. The C-Flag gets the last bit shifted out of REG bit 0. Bits shifted out of the C-Flag are lost.

INDICATORS:                N = Set if bit 7 of REG is set  
                             Z = Set if REG = 0  
                             V = Set to exclusive or of N and C flags  
                             C = Set to the value of the last bit shifted out of REG bit 0

---

SSLA                      SHIFT SINGLE LEFT ARITHMETIC

---

FORMAT:                      SSLA     REG, COUNT  
FUNCTION:                    A 17-bit left arithmetic shift is done stored count+1 times on C-Flag:REG. Zeros are shifted into REG bit 0, and the C-FLAG gets the last bit shifted out of REG bit 15. Bits shifted out of the C-Flag are lost.

INDICATORS:                N = Set if REG bit 15 is set  
                             Z = Set if REG = 0  
                             V = Set to exclusive or of N and C flags  
                             C = Set to the value of the last bit shifted out of REG bit 15

---

SDRR                      SHIFT DOUBLE RIGHT ROTATE

---

FORMAT:                      SDRR     REG, COUNT  
FUNCTION:                    REG+1:REG:C-Flag is rotate right stored count+1 times. The C-Flag is shifted into REG+1 bit 15, REG+1 bit 0 is shifted into REG bit 15, and REG bit 0 is shifted into the C-Flag.

INDICATORS:                N = Set if bit 7 of REG is set  
                             Z = Set if REG = 0  
                             V = Set to exclusive or of N and C flags  
                             C = Set to the value of the last bit shifted out of REG bit 0

---

SDLR                      SHIFT DOUBLE LEFT ROTATE

---

FORMAT:                      SDLR     REG, COUNT  
FUNCTION:                    A 33 bit left rotate is done stored count+1 times on C-Flag:REG+1:REG. The C-Flag is shifted into REG bit 0, REG bit 15 is shifted into REG+1 bit 0, and REG+1 bit 15 is shifted into the C-Flag

INDICATORS:                N = Set if REG+1 bit 15 is set  
                             Z = Set if REG+1 = 0  
                             V = Set to exclusive or of N and C flags  
                             C = Set to the value of the last bit shifted out of REG+1 bit 15.

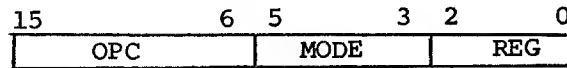
# SHIFT DOUBLE RIGHT ARITHMETIC

SHIFT DOUBLE LEFT ARITHMETIC

17

## FORMAT 7 OP CODES

SINGLE OPS - ONE OR TWO WORDS - DM0 TO DM7



There are 32 op codes in this class representing op codes "0A00" to "0DFF" and "8A00" to "8DFF". All addressing modes from 0 to 7 are available with all registers available as index registers (see chapter two). A one word op code is generated for addressing modes 0 to 5. A two word op code is generated for addressing modes 6 and 7 with the offset value in word two. For DM6 and DM7 with PC as the index register PC is added to the offset from word two after the offset is fetched from memory. The offset is therefore relative to a PC that points to the op code that follows (i.e. current op code + 4). Codes "8A00" to "8CC0" are BYTE ops.

BASE OP CODE	MNEMONIC	BASE OP CODE	MNEMONIC
0A00	ROR	8A00	RORB
0A40	ROL	8A40	ROLB
0A80	TST	8A80	TSTB
0AC0	ASL	8AC0	ASLB
0B00	SET	8B00	SETB
0B40	CLR	8B40	CLRB
0B80	ASR	8B80	ASRB
0BC0	SWAB	8BC0	SWAD
0C00	COM	8C00	COMB
0C40	NEG	8C40	NEGB
0C80	INC	8C80	INCB
0CC0	DEC	8CC0	DECB
0D00	IW2	8D00	LSTS
0D40	SXT	8D40	SSTS
0D80	TCALL	8D80	ADC
0DC0	TJMP	8DC0	SBC

### WORD OPS

---

ROR                      ROTATE RIGHT

---

FORMAT:                      ROR     DST

FUNCTION:                    A 1-bit right rotate is done on (DST):C-Flag  
The C-Flag is shifted into (DST) bit 15, and (DST) bit 0 is shifted into the C-flag.

INDICATORS:                N = Set if bit 7 of (DST) is set  
Z = Set if (DST) = 0  
V = Set to exclusive or of N and C flags  
C = Set to the value of the bit shifted out of (DST)

---

ROL                      ROTATE LEFT

---

FORMAT:                      ROL     DST

FUNCTION:                    A 1-bit left rotate is done on C-Flag:(DST). The

C-Flag is shifted into (DST) bit 0, and (DST) bit 15 is shifted into the C-Flag.

INDICATORS: N = Set if bit 15 of (DST) is set  
 Z = Set if (DST) = 0  
 V = Set to exclusive or of N and C flags  
 C = Set to the value of the bit shifted out of (DST)

---

TST TEST WORD

---

FORMAT: TST DST  
 OPERATION: (DST)  $\Delta$  (DST)  
 FUNCTION: The indicators are set to reflect the destination operand status.  
 INDICATORS: N = Set if (DST) bit 15 is set  
 Z = Set if (DST) = 0  
 V = Reset  
 C = Unchanged

---

ASL ARITHMETIC SHIFT LEFT

---

FORMAT: ASL DST  
 FUNCTION: A 1-bit left arithmetic shift is done on (DST). A zero is shifted into (DST) bit 0, and (DST) bit 15 is shifted into the C-Flag.  
 INDICATORS: N = Set if (DST) bit 15 is set  
 Z = Set if (DST) = 0  
 V = Set to exclusive or of N and C flags  
 C = Set to the value of the bit shifted out of (DST)

---

SET SET TO ONES

---

FORMAT: SET DST  
 OPERATION: (DST)  $\leftarrow$  "FFFF"  
 FUNCTION: The destination operand is set to all ones  
 INDICATORS: N = Set  
 Z = Reset  
 V = Reset  
 C = Unchanged

---

CLR CLEAR TO ZEROS

---

FORMAT: CLR DST  
 OPERATION: (DST)  $\leftarrow$  0  
 FUNCTION: The destination operand is cleared to all zeros  
 INDICATORS: N = Reset  
 Z = Set  
 V = Reset  
 C = Unchanged if DM0. Reset if DM1-DM7.

---

ASR ARITHMETIC SHIFT RIGHT

---

FORMAT: ASR DST  
 FUNCTION: A 1-bit right arithmetic shift is done on (DST). Bit 15 of (DST) is replicated. Bit 0 of (DST) is shifted into the C-Flag.



INDICATORS: N = Set if (DST) bit 7 is set  
 Z = Set if (DST) =  $\emptyset$   
 V = Set to exclusive or of N and C flags  
 C = Set to the value of the bit shifted out of (DST)

---

SWAB SWAP BYTES

---

FORMAT: SWAB DST  
 OPERATION: (DST) 15-8  $\leftrightarrow$  (DST) 7- $\emptyset$   
 FUNCTION: The upper and lower bytes of (DST) are exchanged.  
 INDICATORS: N = Set if (DST) bit 7 is set  
 Z = Set if (DST) lower byte =  $\emptyset$   
 V = Reset  
 C = Unchanged

---

COM COMPLEMENT

---

FORMAT: COM DST  
 OPERATION: (DST)  $\leftarrow$  (DST)  
 FUNCTION: The destination operand is one's complemented.  
 INDICATORS: N = Set if (DST) bit 15 is set  
 Z = Set if (DST) =  $\emptyset$   
 V = Reset  
 C = Set

---

NEG NEGATE

---

FORMAT: NEG DST  
 OPERATION: (DST)  $\leftarrow$  -(DST)  
 FUNCTION: The destination operand is two's complemented.  
 INDICATORS: N = Set if (DST) bit 15 is set  
 Z = Set if (DST) =  $\emptyset$   
 V = Set if (DST) = "8 $\emptyset\emptyset\emptyset$ "  
 C = Reset if (DST) =  $\emptyset$

---

INC INCREMENT

---

FORMAT: INC DST  
 OPERATION: (DST)  $\leftarrow$  (DST) + 1  
 FUNCTION: The destination operand is incremented by one.  
 INDICATORS: N = Set if (DST) bit 15 is set  
 Z = Set if (DST) =  $\emptyset$   
 V = Set if (DST) = "8 $\emptyset\emptyset\emptyset$ "  
 C = Set if a carry is generated from (DST) bit 15

---

DEC DECREMENT

---

FORMAT: DEC DST  
 OPERATION: (DST)  $\leftarrow$  (DST) - 1  
 FUNCTION: The destination operand is decremented by one.  
 INDICATORS: N = Set if (DST) bit 15 is set  
 Z = Set if (DST) =  $\emptyset$   
 V = Set if (DST) = "7FFF"  
 C = Set if a borrow is generated from (DST) bit 15

---

IW2	INCREMENT WORD BY TWO
-----	-----------------------

---

FORMAT:	IW2     DST
OPERATION:	$(DST) \leftarrow (DST) + 2$
FUNCTION:	The destination operand is incremented by two.
INDICATORS:	N = Set if (DST) bit 15 is set Z = Set if (DST) = 0 V = Set if (DST) = "8000" or "8001" C = Set if a carry is generated from (DST) bit 15

---

SXT	SIGN EXTEND
-----	-------------

---

FORMAT:	SXT     DST
OPERATION:	IF N = 0, $(DST) \leftarrow 0$ IF N = 1, $(DST) \leftarrow \text{"FFFF"}$
FUNCTION:	The N-Flag status is replicated in the destination operand
INDICATORS:	Unchanged

---

TCALL	TABLED SUBROUTINE CALL
-------	------------------------

---

FORMAT:	TCALL     DST
OPERATION:	$\downarrow SP, @SP \leftarrow PC$ $PC \leftarrow PC + (DST)$ $PC \leftarrow PC + @PC$
FUNCTION:	PC, which points to the op code that follows, is pushed onto the stack. The destination operand is added to PC. The contents of this intermediate table address is also added to PC to get the final destination address. Note that at least one op code must exist between the TCALL and the table for a subroutine return.
INDICATORS:	Unchanged

---

TJMP	TABLED JUMP
------	-------------

---

FORMAT:	TJMP     DST
OPERATION:	$PC \leftarrow PC + (DST)$ $PC \leftarrow PC + @PC$
FUNCTION:	The destination operand is added to PC, and the contents of this intermediate location is also added to PC to get the final destination address.
INDICATORS:	Unchanged

---

LSTS	LOAD PROCESSOR STATUS
------	-----------------------

---

FORMAT:	LSTS     DST
FUNCTION:	The four indicators and the interrupt enable (I2) are loaded from the destination operand.
INDICATORS:	Set to the status of (DST) bits 0 - 3

---

SSTS	STORE PROCESSOR STATUS
------	------------------------

---

FORMAT:	SSTS     DST
FUNCTION:	The processor status word is formed and stored in (DST).
INDICATORS:	Unchanged

<u>ADC</u>	<u>ADD CARRY</u>
FORMAT:	ADC     DST
OPERATION:	$(DST) \leftarrow (DST) + C\text{-flag}$
FUNCTION:	The carry flag is added to the destination operand
INDICATORS:	N = Set if (DST) bit 15 is set Z = Set if $(DST) = \emptyset$ V = Set to exclusive or of N and C flags C = Set if a carry is generated from (DST) bit 15

<u>SBC</u>	<u>SUBTRACT CARRY</u>
FORMAT:	SBC     DST
OPERATION:	$(DST) \leftarrow (DST) - C\text{-Flag}$
FUNCTION:	The Carry flag is subtracted from the destination operand
INDICATORS:	N = Set if (DST) bit 15 is set Z = Set if $(DST) = \emptyset$ V = Set to exclusive or of N and C flags C = Set if a borrow is generated from (DST) bit 15

#### BYTE OPS

For DM $\emptyset$  addressing only the lower byte of the destination register is affected by a byte op code. For DM1-DM7 addressing only the specified memory byte is affected by a byte op. For even memory addresses the lower byte is altered, and for odd memory addresses the upper byte is altered.

<u>RORB</u>	<u>ROTATE RIGHT BYTE</u>
FORMAT:	RORB     DST
FUNCTION:	A 1-bit right rotate is done on $(DST)_B$ :C-Flag. Bit $\emptyset$ of $(DST)_B$ is shifted into the C-Flag, and the C-Flag is shifted into $(DST)_B$ bit 7.
INDICATORS:	N = Set if $(DST)_B$ bit 7 is set Z = Set if $(DST)_B = \emptyset$ V = Set to exclusive or of N and C flags C = Set to the value of the bit shifted out of $(DST)_B$ bit $\emptyset$

<u>ROLB</u>	<u>ROTATE LEFT BYTE</u>
FORMAT:	ROLB     DST
FUNCTION:	A 1-bit left rotate is done on C-flag : $(DST)_B$ . Bit 7 of $(DST)_B$ is shifted into the C-flag, and the C-flag is shifted into $(DST)_B$ bit $\emptyset$
INDICATORS:	N = Set if $(DST)_B$ bit 7 is set Z = Set if $(DST)_B = \emptyset$ V = Set to exclusive or of N and C flags C = Set to the value of the bit shifted out of $(DST)_B$ bit 7

<u>TSTB</u>	<u>TEST BYTE</u>
FORMAT:	TSTB     DST
OPERATION:	$(DST)_B \Delta (DST)_B$

FUNCTION: The destination operand status sets the indicators.  
 INDICATORS: N = Set if (DST)<sub>B</sub> bit 7 is set  
 Z = Set if (DST)<sub>B</sub> = 0  
 V = Reset  
 C = Unchanged

---

ASLB ARITHMETIC SHIFT LEFT BYTE

---

FORMAT: ASLB DST  
 FUNCTION: A 1-bit left arithmetic shift is done on C-Flag:(DST)<sub>B</sub>.  
 A zero is shifted into (DST)<sub>B</sub> bit 0, and (DST)<sub>B</sub> bit 7 is  
 shifted into the C-flag.  
 INDICATORS: N = set if (DST)<sub>B</sub> bit 7 is set  
 Z = Set if (DST)<sub>B</sub> = 0  
 V = Set to exclusive or of N and C flags  
 C = Set to the value of the bit shifted out of (DST)<sub>B</sub> bit 7

---

SETB SET BYTE TO ONES

---

FORMAT: SETB DST  
 OPERATION: (DST)<sub>B</sub> ← "FF"  
 FUNCTION: The destination byte operand is set to all ones  
 INDICATORS: N = Set  
 Z = Reset  
 V = Reset  
 C = Unchanged

---

CLRB CLEAR BYTE TO ZEROS

---

FORMAT: CLRB DST  
 OPERATION: (DST)<sub>B</sub> ← 0  
 FUNCTION: The destination byte operand is cleared to all zeros.  
 INDICATORS: N = Reset  
 Z = Set  
 V = Reset  
 C = Reset

---

ASRB ARITHMETIC SHIFT RIGHT BYTE

---

FORMAT: ASRB DST  
 FUNCTION: A 1-bit right arithmetic shift is done on (DST)<sub>B</sub>:  
 C-flag. Bit 7 of (DST)<sub>B</sub> is replicated. Bit 0 of  
 (DST)<sub>B</sub> is shifted into the C-flag.  
 INDICATORS: N = Set if (DST)<sub>B</sub> bit 7 is set  
 Z = Set if (DST)<sub>B</sub> = 0  
 V = Set to exclusive or of N and C flags  
 C = Set to the value of the bit shifted out of (DST)<sub>B</sub> bit 0

---

SWAD SWAP DIGITS

---

FORMAT: SWAD DST  
 FUNCTION: The two hex digits in the destination byte operand  
 are exchanged with each other,  
 INDICATORS: N = Set if (DST)<sub>B</sub> bit 7 is set  
 Z = Set if (DST)<sub>B</sub> = 0  
 V = Set if (DST)<sub>B</sub> bit 7 is set  
 C = Reset

---

COMB                      COMPLEMENT BYTE

---

FORMAT:                  COMB      DST  
OPERATION:                 $(DST)_B \leftarrow \overline{(DST)_B}$   
FUNCTION:                The destination byte operand is one's complemented  
INDICATORS:              N = Set if  $(DST)_B$  bit 7 is set  
                          Z = Set if  $(DST)_B = \emptyset$   
                          V = Reset  
                          C = Set

---

NEGB                      NEGATE BYTE

---

FORMAT:                  NEGB      DST  
OPERATION:                 $(DST)_B \leftarrow -(DST)_B$   
FUNCTION:                The destination byte operand is two's complemented  
INDICATORS:              N = Set if  $(DST)_B$  bit 7 is set  
                          Z = Set if  $(DST)_B = \emptyset$   
                          V = Set if  $(DST)_B = "8\emptyset\emptyset\emptyset"$   
                          C = Reset if  $(DST)_B = \emptyset$

---

INCB                      INCREMENT BYTE

---

FORMAT:                  INCB      DST  
OPERATION:                 $(DST)_B \leftarrow (DST)_B + 1$   
FUNCTION:                The destination byte operand is incremented by one  
INDICATORS:              N = Set if  $(DST)_B$  is set  
                          Z = Set if  $(DST)_B = \emptyset$   
                          V = Set if  $(DST)_B = "8\emptyset\emptyset\emptyset"$   
                          C = Set if a carry is generated from  $(DST)_B$  bit 7

---

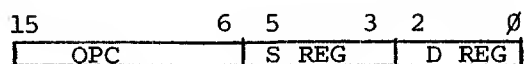
DECB                      DECREMENT BYTE

---

FORMAT:                  DECB      DST  
OPERATION:                 $(DST)_B \leftarrow (DST)_B - 1$   
FUNCTION:                The destination byte operand is decremented by one  
INDICATORS:              N = Set if  $(DST)_B$  bit 7 is set  
                          Z = Set if  $(DST)_B = \emptyset$   
                          V = Set if  $(DST)_B = "7FFF"$   
                          C = Set if a borrow is generated from  $(DST)_B$  bit 7

# FORMAT 8 OP CODES\*

DOUBLE OPS - SINGLE WORD - SMØ AND DMØ ONLY



There are 8 op codes in this class representing op codes "ØEØØ" to "ØFFF". Only addressing mode Ø is allowed for both the source and destination. All are one word op codes, and all are block move instructions. The last 4 can be used as pseudo DMA ops in some hardware configurations. In all cases the source register contains the address of the first word or byte of memory to be moved, and the destination register contains the address of the first word or byte of memory to receive the data being moved. The number of words or bytes being moved is contained in RØ. The count ranges from 1-65536 (Ø = 65536) words or bytes. The count in RØ is an unsigned positive integer. None of the indicators are altered by these op codes.

Each of these op codes is interruptable at the end of each word or byte transfer. If no interrupt requests are active the transfers continue. PC is not incremented to the next op code until the op code is completed. This allows for complete interruptability as long as register integrity is maintained during the interrupt.

## BASE OP CODE                      MNEMONIC

ØEØØ	MBWU
ØE4Ø	MBWD
ØE8Ø	MBBU
ØECØ	MBBD
ØFØØ	MBWA
ØF4Ø	MBBA
ØF8Ø	MABW
ØFCØ	MABB

\* NOTE: These op codes are all in the third microm.

## MBWU                                      MOVE BLOCK OF WORDS UP

FORMAT: MBWU SRC, DST  
 FUNCTION: The word string beginning with the word addressed by the source register is moved to successively increasing word addresses as specified by the destination register. The source and destination registers are each incremented by two after each word is transferred. RØ is decremented by one after each transfer, and transfers continue until RØ = Ø.

## MBWD                                      MOVE BLOCK OF WORDS DOWN

FORMAT: MBWD SRC, DST  
 FUNCTION: The word string beginning with the word addressed by the source register is moved to successively

decreasing word addresses as specified by the destination register. The source and destination registers are each decremented by two after each word is transferred. RØ is decremented by one after each transfer, and transfers continue until RØ = Ø.

INDICATORS:

Unchanged

MBBU

MOVE BLOCK OF BYTES UP

FORMAT:

MBBU SRC, DST

FUNCTION:

The byte string beginning with the byte addressed by the source register is moved to successively increasing byte addresses as specified by the destination register. The source and destination registers are each incremented by one after each byte is transferred. RØ is decremented by one after each transfer, and transfers continue until RØ = Ø.

INDICATORS:

Unchanged.

MBBD

MOVE BLOCK OF BYTES DOWN

FORMAT:

MBBD SRC, DST

FUNCTION:

The byte string beginning with the byte addressed by the source register is moved to successively decreasing byte addresses as specified by the destination register. The source register, destination register, and RØ, are each decremented by one after each byte is transferred. Transfers continue until RØ = Ø.

INDICATORS:

Unchanged

MBWA

MOVE BLOCK OF WORDS TO ADDRESS

FORMAT:

MBWA SRC, DST

FUNCTION:

Same as MBWU except that the destination register is never incremented.

INDICATORS:

Unchanged

MBBA

MOVE BLOCK OF BYTES TO ADDRESS

FORMAT:

MBBA SRC, DST

FUNCTION:

Same as MBBU except that the destination register is never incremented.

INDICATORS:

Unchanged

MABW

MOVE ADDRESS TO BLOCK OF WORDS

FORMAT:

MABW SRC, DST

FUNCTION:

Same as MBWU except that the source register is never incremented.

INDICATORS:

Unchanged

MABB

MOVE ADDRESS TO BLOCK OF BYTES

FORMAT:

MABB SRC, DST

FUNCTION:

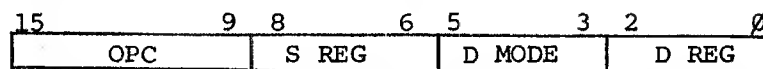
Same as MBBU except that the source register is never incremented.

INDICATORS:

Unchanged

## FORMAT 9 OP CODES

DOUBLE OPS - ONE OR TWO WORDS - SM $\emptyset$ , DM $\emptyset$  to DM7



There are 8 op codes in this class representing op codes "7 $\emptyset\emptyset\emptyset$ " to "7FFF". Source mode  $\emptyset$  addressing only is allowed, but destination modes  $\emptyset$  - 7 are allowed for all op codes except 3: JSR and LEA with DM $\emptyset$  will cause an illegal instruction format trap (see chapter 2), and SOB is a special format unique to itself. It is included here only because its destination field is 6 bits long. SOB is a branch instruction. Its 6 bit destination field is a positive word offset from PC, which points to the op code that follows, backwards to the desired address. Forward branching is not allowed. SOB is always a one word op code, and it is used for fast loop control. All other op codes are one word long for DM $\emptyset$  to DM5 addressing and two words long for DM6 or DM7 addressing. The rules for PC relative addressing with DM6 or DM7 are the same as they are for the format 7 op codes. Preliminary decoding of all these op codes except SOB presets the indicator flags as follows: N = 1, Z =  $\emptyset$ , V =  $\emptyset$ , C = 1.

BASE OP CODE	MNEMONIC
--------------	----------

7 $\emptyset\emptyset\emptyset$	JSR
72 $\emptyset\emptyset$	LEA
74 $\emptyset\emptyset$	ASH
76 $\emptyset\emptyset$	SOB
78 $\emptyset\emptyset$	XCH
7A $\emptyset\emptyset$	ASHC
7C $\emptyset\emptyset$	MUL
7E $\emptyset\emptyset$	DIV

JSR	JUMP TO SUBROUTINE
-----	--------------------

FORMAT:	JSR    REG, DST
OPERATION:	↓ SP, @SP ← REG REG ← PC PC ← DST
FUNCTION:	The linkage register is pushed onto the stack; PC, which points to the op code that follows, is placed in the linkage register; and the destination address is placed in PC. DM $\emptyset$ is illegal. The assembler recognizes the format "CALL DST" as being equivalent to "JSR    PC, DST".
INDICATORS:	Preset

LEA	LOAD EFFECTIVE ADDRESS
-----	------------------------

FORMAT:	LEA    REG, DST
OPERATION:	REG ← DST



FUNCTION:	The destination address is placed into the source register. DMØ is illegal. The assembler recognizes the format "JMP DST" as being equivalent to "LEA PC,DST".
INDICATORS:	Preset
<u>XCH</u>	<u>EXCHANGE</u>
FORMAT:	XCH REG, DST
OPERATION:	REG $\nleftrightarrow$ (DST)
FUNCTION:	The source register and destination contents are exchanged with each other.
INDICATORS:	Preset
<u>SOB</u>	<u>SUBTRACT ONE AND BRANCH (IF <math>\neq</math> Ø)</u>
FORMAT:	SOB REG, DST
OPERATION:	REG $\leftarrow$ REG - 1 IF REG $\neq$ Ø, PC $\leftarrow$ PC - (OFFSET *2)
FUNCTION:	The source register is decremented by one. If the result is not zero then twice the value of the destination offset is subtracted from PC.
INDICATORS:	Unchanged
<u>ASH</u>	<u>ARITHMETIC SHIFT</u>
FORMAT:	ASH REG, DST
FUNCTION:	The source register is shifted arithmetically with the number of bits and direction specified by the destination operand. If (DST) = Ø no shifting occurs. If (DST) = -X then REG is shifted right arithmetically X bits as in an SSRA. If (DST) = +X then REG is shifted left arithmetically X bits as in an SSLA. Only an 8 bit destination operand is used. Thus, DST is a byte address. For DMØ only the lower byte of the destination register is used.
INDICATORS:	Preset if (DST) = Ø. Otherwise: N = Set if REG bit 15 is set Z = Set if REG = Ø V = Set to exclusive or of N and C flags C = Set to the value of the last bit shifted out of REG
<u>ASHC</u>	<u>ARITHMETIC SHIFT COMBINED</u>
FORMAT:	ASHC REG, DST
FUNCTION:	Exactly the same as ASH except that the shift is done on REG+1:REG. All other comments apply.
INDICATORS:	Preset if (DST) = Ø. Otherwise: N = Set if REG+1 bit 15 is set Z = Set if REG+1: REG = Ø V = Reset C = Set to the value of the last bit shifted out

MULMULTIPLY

FORMAT: MUL REG, DST  
OPERATION:  $REG+1:REG \leftarrow REG * (DST)$   
FUNCTION: An unsigned multiply is performed on the source register and the destination operand. The unsigned 32 bit result is placed in REG+1:REG.  
INDICATORS: N = Set if REG+1 bit 15 is set  
Z = Set if  $REG+1:REG = \emptyset$   
V = Reset  
C = Indeterminate

DIVDIVIDE

FORMAT: DIV REG, DST  
OPERATION:  $REG \leftarrow [REG+1:REG / (DST)]$   
 $REG+1 \leftarrow \text{REMAINDER}$   
FUNCTION: An unsigned divide is performed on the 32 bit source operand REG+1:REG and the destination operand. The unsigned result is placed in REG, and the unsigned remainder is placed in REG+1. No divide occurs and the V-flag is set if REG+1 is greater than or equal to (DST) since the result will not fit into 16 bits. If the divisor is zero both the V and C flags are set.  
INDICATORS: If no division error:  
N = Set if REG bit 15 is set  
Z = Set if  $REG = \emptyset$   
V = Reset  
C = Indeterminate  
If division error:  
N = Reset  
Z = Reset  
V = Set  
C = set if  $(DST) = \emptyset$

## FORMAT 10 OP CODES

DOUBLE OPS - ONE TO THREE WORDS - SM $\emptyset$  TO SM7, DM $\emptyset$  TO DM7.

15	12	11	9	8	6	5	3	2	0
OPC			S MODE		S REG		D MODE		D REG

There are 12 op codes in this class representing op codes "1 $\emptyset\emptyset\emptyset$ " to "6FFF" and "9 $\emptyset\emptyset\emptyset$ " to "EFFF". Nine of the op codes are word ops. Three are byte ops. Full source and destination mode addressing with any register is allowed. A one word op code is generated for SM $\emptyset$ -SM5 and DM $\emptyset$ -DM5 addressing. A two word op code is generated for either SM6-SM7 or DM6-DM7 addressing, but not both. For both SM6-SM7 and DM6-DM7 addressing a three word op code is generated. For a two word op code with word #1 at location X: X + 2 contains the source or destination offset and PC = X + 4 if PC is the register that applies to the offset in location X + 2. For a three word op code with word #1 at location X: X + 2 contains the source offset and X + 4 contains the destination offset. If the source register is PC then PC = X + 4 when added to the offset to compute the source address. If the destination register is PC then PC = X + 6 when added to the offset to compute the destination address.

BASE OP CODE	MNEMONIC
1 $\emptyset\emptyset\emptyset$	ADD
2 $\emptyset\emptyset\emptyset$	SUB
3 $\emptyset\emptyset\emptyset$	AND
4 $\emptyset\emptyset\emptyset$	BIC
5 $\emptyset\emptyset\emptyset$	BIS
6 $\emptyset\emptyset\emptyset$	XOR
9 $\emptyset\emptyset\emptyset$	CMP
A $\emptyset\emptyset\emptyset$	BIT
B $\emptyset\emptyset\emptyset$	MOV
C $\emptyset\emptyset\emptyset$	CMPB
D $\emptyset\emptyset\emptyset$	MOVB
E $\emptyset\emptyset\emptyset$	BISB

### WORD OPS

ADD	ADD
FORMAT:	ADD SRC, DST
OPERATION:	(DST) $\leftarrow$ (SRC) + (DST)
FUNCTION:	The source and destination operands are added together, and the sum is placed in the destination.
INDICATORS:	N = Set if (DST) bit 15 is set Z = Set if (DST) = 0 V = Set if both operands were of the same sign and the result was of the opposite sign C = Set if a carry is generated from bit 15 of the result

---

SUB                      SUBTRACT

---

FORMAT:                SUB     SRC, DST  
OPERATION:              $(DST) \leftarrow (DST) - (SRC)$   
FUNCTION:              The two's complement of the source operand is added to the destination operand, and the sum is placed in the destination.  
INDICATORS:           N = Set if (DST) bit 15 is set  
                      Z = Set if (DST) = 0  
                      V = Set if operands were of different signs and the sign of the result is the same as the sign of the source operand  
                      C = Set if a borrow is generated from bit 15 of the result

---

AND                     AND

---

FORMAT:                AND     SRC, DST  
OPERATION:              $(DST) \leftarrow (SRC) \wedge (DST)$   
FUNCTION:              The source and destination operands are logically AND'ed together, and the result is placed in the destination.  
INDICATORS:           N = Set if (DST) bit 15 is set  
                      Z = Set if (DST) = 0  
                      V = Reset  
                      C = Unchanged

---

BIC                     BIT CLEAR

---

FORMAT:                BIC     SRC, DST  
OPERATION:              $(DST) \leftarrow (SRC) \wedge (DST)$   
FUNCTION:              The one's complement of the source operand is logically AND'ed with the destination operand, and the result is placed in the destination.  
INDICATORS:           N = Set if (DST) bit 15 is set  
                      Z = Set if (DST) = 0  
                      V = Reset  
                      C = Unchanged

---

BIS                     BIT SET

---

FORMAT:                BIS     SRC, DST  
OPERATION:              $(DST) \leftarrow (SRC) \vee (DST)$   
FUNCTION:              The source and destination operands are logically OR'ed, and the result is placed in the destination.  
INDICATORS:           N = Set if (DST) bit 15 is set  
                      Z = Set if (DST) = 0  
                      V = Reset  
                      C = Unchanged

---

XOR                     EXCLUSIVE OR

---

FORMAT:                XOR     SRC, DST  
OPERATION:              $(DST) \leftarrow (SRC) \oplus (DST)$   
FUNCTION:              The source and destination operands are logically EXCLUSIVE OR'ed, and the result is placed in the destination.

INDICATORS:        N = Set if (DST) bit 15 is set  
                      Z = Set if (DST) =  $\emptyset$   
                      V = Reset  
                      C = Unchanged

---

**CMP                      COMPARE**

---

FORMAT:            CMP     SRC, DST  
 OPERATION:        (SRC) - (DST)  
 FUNCTION:         The destination operand is subtracted from the  
                      source operand, and the result sets the indicators. Neither operand is altered.  
 INDICATORS:       N = Set if result bit 15 is set  
                      Z = Set if result =  $\emptyset$   
                      V = Set if operands were of opposite sign and the  
                      sign of the result is the same as the sign of (DST)  
                      C = Set if a borrow is generated from bit 15 of the  
                      result

---

**BIT                      BIT TEST**

---

FORMAT:            BIT     SRC, DST  
 OPERATION:        (SRC)  $\wedge$  (DST)  
 FUNCTION:         The source and destination operands are logically  
                      ANDED, and the result sets the indicators. Neither  
                      operand is altered.  
 INDICATORS:       N = Set if result bit 15 is set  
                      Z = Set if result =  $\emptyset$   
                      V = Reset  
                      C = Unchanged

---

**MOV                      MOVE**

---

FORMAT:            MOV     SRC, DST  
 OPERATION:        (DST)  $\leftarrow$  (SRC)  
 FUNCTION:         The destination operand is replaced with the source  
                      operand.  
 INDICATORS:       N = Set if (DST) bit 15 is set  
                      Z = Set if (DST) =  $\emptyset$   
                      V = Reset  
                      C = Unchanged

BYTE OPS

For SM $\emptyset$  addressing only the lower byte of the source register is used as an operand. For SM1-SM7 addressing only the addressed memory byte is used as an operand. For DM $\emptyset$  addressing only the lower byte of the destination register is used as an operand with one exception: MOV<sub>B</sub> will extend the sign through bit 15. For DM1-DM7 addressing only the addressed memory byte is used as an operand.

---

**CMPB                    COMPARE BYTE**

---

FORMAT:            CMPB    SRC, DST  
 OPERATION:        (SRC)<sub>B</sub> - (DST)<sub>B</sub>

**FUNCTION:** The destination operand is subtracted from the source operand, and the result sets the indicators. Neither operand is altered.  
**INDICATORS:** N = Set if result bit 7 is set  
 Z = Set if result =  $\emptyset$   
 V = Set if operands were of different signs and the sign of the result is the same as the sign of (DST)B.  
 C = Set if a borrow is generated from result bit 7

---

**MOVB                      MOVE BYTE**

---

**FORMAT:**                MOVB     SRC, DST  
**OPERATION:**            (DST)B  $\leftarrow$  (SRC)B  
**FUNCTION:**            The destination operand is replaced with the source operand. If DM $\emptyset$  the sign bit (bit 7) is replicated through bit 15.  
**INDICATORS:**        N = Set if (DST)B bit 7 is set  
                       Z = Set if (DST)B =  $\emptyset$   
                       V = Reset  
                       C = Unchanged

---

**BISB                      BIT SET BYTE**

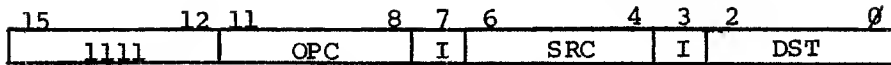
---

**FORMAT:**                BISB     SRC, DST  
**OPERATION:**            (DST)B  $\leftarrow$  (SRC)B  $\vee$  (DST)B  
**FUNCTION:**            The source and destination operands are logically ORED, and the result is placed in the destination.  
**INDICATORS:**        N = Set if (DST)B bit 7 is set  
                       Z = Set if (DST)B =  $\emptyset$   
                       V = Reset  
                       C = Unchanged

When using auto increments or decrements in either the source or destination (or both) fields the user must remember the following rule: All increments or decrements in the source are fully completed before any destination decoding begins even if the same index register is used in both the source and destination. The two fields are totally independent.

## FORMAT 11 OP CODES

DOUBLE OPS - ONE WORD - FLOATING POINT.



There are 16 OP Codes in this class representing OP Codes "F000" to "FFFF". Only five are currently defined. They reside in the third microm along with the Format 8 OP Codes. The remaining 11 OP Codes are mapped to the fourth microm for future expansion or customized user OP Codes. All are one word long. Two source and destination addressing modes are available. These two modes, FP0 and FP1, are unique to these OP Codes. Each consists of a 3-bit Register Designation and a 1 bit indirect flag preceeding the register designator. For FP0 the indirect bit is 0, and FP1 it is one. Both the source and destination fields have both addressing modes. The modes are defined as follows:

FP0     The designated register contains the address of the operand.

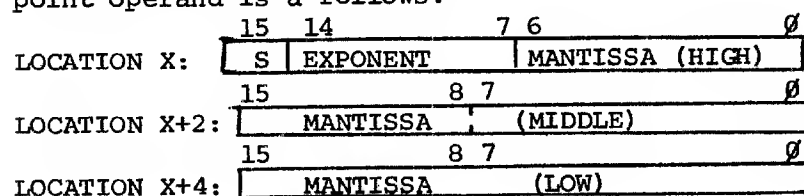
FP1     The designated register contains the address of the address of the operand.

FP0     is the same as standard addressing mode 1, and FP1 is the same as standard addressing mode 7 with an offset of zero.

The computed address is the address of the first word of a 3 word floating point operand. The first word contains the sign, exponent, and high byte of the mantissa. The next higher address contains the middle two bytes of the mantissa, and the next higher address after that contains the lowest two bytes of the mantissa. This format is half way between single and double precision floating point formats, and it represents the most efficient use of microprocessor ROM and register space. The complete format is as follows:

1. A 1 bit sign for the entire number which is zero for positive.
2. An 8-bit base-two exponent in excess-128 notation with a range of +127, -128. The only legal number with an exponent of -128 is true zero (all zeros).
3. A 40 bit mantissa with the MSB implied.

Since every operand is assumed to be normalized upon entry and every result is normalized before storage in the destination addresses, and since a normalized mantissa has a MSB equal to one, then only 39 bits need to be stored. The MSB is implied to be a one, and the bit position it normally occupies is taken over by the exponent to increase its range by a factor of two. The full format of a floating point operand is as follows:



True zero is represented by a field of 48 zeroes. In effect, the CPU considers any number with an exponent of all zeroes (-128) to be a zero during multiplication and division. For add and subtract the only legal number with an exponent of -128 is true zero. All others cause erroneous results. No registers are modified by any Format 11 OP Code. However, to make room internally for computations 4 registers are saved in memory locations "30" - "38" during the execution of FADD, FSUB, FMUL and FDIV. These registers are retrieved at the completion of the OP Codes. The registers saved are: the destination address, SP, PC and R0. No Format 11 OP Code is interruptable (for obvious reasons). FMUL uses location "38" for temporary storage of partial results.

#### FLOATING POINT ERROR TRAPS

Location "3E" is defined as the floating point error trap PC. Whenever an overflow, underflow, or divide by zero occurs a standard trap call is executed with PS and PC pushed onto the stack, and PC fetched from location "3E". I2 is not altered. The remaining memory locations that are reserved for the floating point option ("3A and "3C") are not currently used. The status of the indicator flags and destination addresses during the 3 trap conditions are defined as follows:

##### FOR UNDERFLOW (FADD, FSUB, FMUL, FDIV)

N = 1            Destination contains all zeroes  
 Z = 0            (true zero).  
 V = 1  
 C = 0

##### FOR OVERFLOW (FADD, FSUB, FMUL)

N = 0            Destination not altered in any way.  
 Z = 0  
 V = 1  
 C = 0

##### FOR OVER FLOW (FDIV)

N = 0            Destination not altered if overflow detected  
 Z = 0            during exponent computation. Undefined  
 V = 1            otherwise. (Used to save unnormalized  
 C = 0            partial results during a divide).

##### FOR DIVIDE BY ZERO (FDIV)

N = 1            Destination not altered in any way.  
 Z = 0  
 V = 1  
 C = 1

#### RESERVED TRAPS

If the third microm is in the system and the fourth is not then the last 11 floating point OP codes are the only ones that will cause a reserved OP code trap if executed. If the third microm is not in the system then all Format 8 and 11 OP Codes will cause a reserved OP code trap if executed. However, since the Format 8 OP Codes are interrupt-



able the PC is not advance until the completion of the moves. In all other cases PC is advanced when the OP Code is fetched. For these reasons the PC that is saved onto the stack will point to the offending OP Code during a reserved OP Code trap if and only if the offending OP Code is a Format 8 OP Code. For the Format 11 OP Codes the saved PC will point to the OP Code that follows the offending OP Code. If the User wishes to identify which OP Code caused the reserved OP Code trap he must not precede a Format 8 OP Code with a Format 11 OP Code or a literal that looks like a Format 11 OP Code.

#### BASE OP CODE

#### MNEMONIC

F000	FADD
F100	FSUB
F200	FMUL
F300	FDIV
F400	FCMP
F500	
F600	
F700	
F800	
F900	
FA00	
FB00	
FC00	
FD00	
FE00	
FF00	

#### FADD

#### FLOATING POINT ADD

FORMAT:	FADD SRC,DST
OPERATION:	$(DST) \leftarrow (DST) + (SRC)$
FUNCTION:	The source and destination operands are added together, normalized, and the result is stored in place of the destination operand.
INDICATORS:	(if no errors) N = Set if the result sign is negative (set). Z = Set if the result is zero V = Reset C = Reset

#### FSUB

#### FLOATING POINT SUBTRACT

FORMAT:	FSUB SRC, DST
OPERATION:	$(DST) \leftarrow (DST) - (SRC)$
FUNCTION:	The source operand is subtracted from the destination operand. The result is normalized and stored in place of the destination operand.

WARNING: THIS OP CODE COMPLEMENTS THE SIGN OF THE SOURCE OPERAND IN MEMORY AND DOES AN FADD.

INDICATORS:	(if no errors) N = Set if the result sign is negative (set) Z = Set if the result is zero.
-------------	--

V = Reset  
C = Reset

<u>FMUL</u>	<u>FLOATING POINT MULTIPLY</u>
FORMAT:	FMUL SRC, DST
OPERATION:	(DST) ← (DST) * (SRC)
FUNCTION:	The source and destination operands are multiplied together, normalized, and the result is stored in place of the destination operand.
INDICATORS:	(if no errors) N = Set if the sign of the result is negative (set). Z = Set if the result is zero V = Reset C = Reset
<u>FDIV</u>	<u>FLOATING POINT DIVIDE</u>
FORMAT:	FDIV SRC, DST
OPERATION:	(DST) ← (DST) / (SRC)
FUNCTION:	The destination operand is divided by the source operand. The result is normalized and stored in place of the destination operand.
INDICATORS:	(if no errors) N = Set if the sign of the result is negative (set). Z = Set if the result is zero V = Reset C = Reset
<u>FCMP</u>	<u>FLOATING POINT COMPARE</u>
FORMAT:	FCMP SRC, DST
OPERATION:	(SRC) - (DST)
FUNCTION:	The destination operand is compared to the source operand, and the indicators are set to allow a <u>SIGNED</u> conditional branch.
INDICATORS:	N = Set if result is negative Z = Set if result is zero V = Set if arithmetic underflow occurs.* C = Set if a borrow is generated. *

\*NOTE: True if first words of both operands are not equal.

CAUTION: The same physical operand may be used as both the source and destination operand for any of the above floating point OP Codes with no abnormal results except two. They are:

- 1) If an error trap occurs the operand will probably be altered.
- 2) An FSUB gives an answer of  $-2x$ , if  $x \neq \emptyset$ , instead of  $\emptyset$ .



# APPENDIX A

## NUMERIC OP CODE TABLE

OP CODE				MNEMONIC
0000	0000	0000	0000	NOP
0000	0000	0000	0001	RESET
0000	0000	0000	0010	IEN
0000	0000	0000	0011	IDS
0000	0000	0000	0100	HALT
0000	0000	0000	0101	XCT
0000	0000	0000	0110	BPT
0000	0000	0000	0111	WFI
0000	0000	0000	1000	RSVC
0000	0000	0000	1001	RRTT
0000	0000	0000	1010	SAVE
0000	0000	0000	1011	SAVS
0000	0000	0000	1100	REST
0000	0000	0000	1101	RRTN
0000	0000	0000	1110	RSTS
0000	0000	0000	1111	RTT
0000	0000	0001	0REG	IAK
0000	0000	0001	1REG	RTN
0000	0000	0010	0REG	MSKO
0000	0000	0010	1REG	PRTN
0000	0000	0011	ARGU	LCC
0000	0000	01AR	GUME	SVCA
0000	0000	10AR	GUME	SVCB
0000	0000	11AR	GUME	SVCC
0000	0001	DISP	LACE	BR
0000	0010	DISP	LACE	BNE
0000	0011	DISP	LACE	BEQ
0000	0100	DISP	LACE	BGE
0000	0101	DISP	LACE	BLT
0000	0110	DISP	LACE	BGT
0000	0111	DISP	LACE	BLE
0000	100R	EG00	VALU	ADDI
0000	100R	EG01	VALU	SUBI
0000	100R	EG10	VALU	BICI
0000	100R	EG11	VALU	MOVI
0000	1010	00MO	DREG	ROR
0000	1010	01MO	DREG	ROL
0000	1010	10MO	DREG	TST
0000	1010	11MO	DREG	ASL
0000	1011	00MO	DREG	SET
0000	1011	01MO	DREG	CLR
0000	1011	10MO	DREG	ASR
0000	1011	11MO	DREG	SWAB
0000	1100	00MO	DREG	COM
0000	1100	01MO	DREG	NEG
0000	1100	10MO	DREG	INC
0000	1100	11MO	DREG	DEC

## OP CODE

## MNEMONIC

0000	1101	00MO	DREG	IW2
0000	1101	01MO	DREG	SXT
0000	1101	10MO	DREG	TCALL
0000	1101	11MO	DREG	TJMP
0000	1110	00SR	CDST	MBWU
0000	1110	01SR	CDST	MBWD
0000	1110	10SR	CDST	MBBU
0000	1110	11SR	CDST	MBBD
0000	1111	00SR	CDST	MBWA
0000	1111	01SR	CDST	MBBA
0000	1111	10SR	CDST	MABW
0000	1111	11SR	CDST	MABB
0001	SRCR	EGDS	TREG	ADD
0010	SRCR	EGDS	TREG	SUB
0011	SRCR	EGDS	TREG	AND
0100	SRCR	EGDT	TREG	BIC
0101	SRCR	EGDT	TREG	BIS
0110	SRCR	EGDS	TREG	XOR
0111	000R	RRDS	TREG	JSR
0111	001R	RRDS	TREG	LEA
0111	010R	RRDS	TREG	ASH
0111	011R	RRDF	FSET	SOB
0111	100R	RRDS	TREG	XCH
0111	101R	RRDS	TREG	ASHC
0111	110R	RRDS	TREG	MUL
0111	111R	RRDS	TREG	DIV
1000	0000	DISP	LACE	BPL
1000	0001	DISP	LACE	BMI
1000	0010	DISP	LACE	BHI
1000	0011	DISP	LACE	BLOS
1000	0100	DISP	LACE	BVC
1000	0101	DISP	LACE	BVS
1000	0110	DISP	LACE	BCC, BHIS
1000	0111	DISP	LACE	BCS, BLO
1000	100R	EG00	VALU	SSRR
1000	100R	EG01	VALU	SSLR
1000	100R	EG10	VALU	SSRA
1000	100R	EG11	VALU	SSLA
1000	1010	00MO	DREG	RORB
1000	1010	01MO	DREG	ROLB
1000	1010	10MO	DREG	TSTB
1000	1010	11MO	DREG	ASLB
1000	1011	00MO	DREG	SETB
1000	1011	01MO	DREG	CLRB
1000	1011	10MO	DREG	ASRB
1000	1011	11MO	DREG	SWAD
1000	1100	00MO	DREG	COMB
1000	1100	01MO	DREG	NEGB
1000	1100	10MO	DREG	INCB
1000	1100	11MO	DREG	DECB

OP CODE				MNEMONIC
1000	1101	00MO	DREG	LSTS
1000	1101	01MO	DREG	SSTS
1000	1101	10MO	DREG	ADC
1000	1101	11MO	DREG	SBC
1000	111R	EG00	VALU	SDRR
1000	111R	EG01	VALU	SDLR
1000	111R	EG10	VALU	SDRA
1000	111R	EG11	VALU	SDLA
1001	SRCR	EGDS	TREG	CMP
1010	SRCR	EGDS	TREG	BIT
1011	SRCR	EGDS	TREG	MOV
1100	SRCR	EGDS	TREG	CMPB
1101	SRCR	EGDS	TREG	MOVB
1110	SRCR	EGDS	TREG	BISB
1111	0000	ISRC	IDST	FADD
1111	0001	ISRC	IDST	FSUB
1111	0010	ISRC	IDST	FMUL
1111	0011	ISRC	IDST	FDIV
1111	0100	ISRC	IDST	FCMP
1111	0101	ISRC	IDST	
1111	0110	ISRC	IDST	
1111	0111	ISRC	IDST	
1111	1000	ISRC	IDST	
1111	1001	ISRC	IDST	
1111	1010	ISRC	IDST	
1111	1011	ISRC	IDST	
1111	1100	ISRC	IDST	
1111	1101	ISRC	IDST	
1111	1110	ISRC	IDST	
1111	1111	ISRC	IDST	



## APPENDIX B

### ASSEMBLER NOTES

#### FORMAT 1 OP CODES

All are one word op codes except SAVS which is a two word op code. The second word of the SAVS op code is an absolute value.

#### FORMAT 2 OP CODES

All are one word with a 3 bit register argument

#### FORMAT 3 OP CODE

A one word op code with a 4 bit numeric argument

#### FORMAT 4 OP CODES

All are one word with a 6 bit numeric argument

#### FORMAT 5 OP CODES

All are one word with an 8 bit signed PC relative word displacement. The displacement is relative to op code+2. Maximum displacement from the op code is +128, -127 words.

#### FORMAT 6 OP CODES

All are one word with a 3 bit register and a 4 bit numeric argument. The stored numeric argument is a positive number from 0 - "F" that equals the actual numeric argument (1-"10") minus one.

#### FORMAT 7 OP CODES

All are one word op codes for DM0 - DM5 addressing and two word op codes for DM6 - DM7 addressing. For DM6 - DM7 addressing the offset is in the second word. If the index register is PC with DM6 - DM7 the offset is relative to op code+4.

#### FORMAT 8 OP CODES

All are one word with a 3 bit source and a 3 bit destination register argument. The count register is implied to be R0.

#### FORMAT 9 OP CODES

All have a 3 bit register argument with a 6 bit destination argument that allows DM0 - DM7 addressing. For DM0 - DM5 a one word op code is generated. For DM6 - DM7 a two word op code is generated with the offset in word two. If the index register is PC with DM6-DM7 then the offset is relative to op code+4.



#### FORMAT 10 OP CODES

All have a 6 bit source and a 6 bit destination argument that allow SM0 - SM7 and DM0 - DM7 addressing. For SM0 - SM5 and DM0 - DM5 combined addressing a one word op code is generated. For SM6 - SM7 or DM6 - DM7 but not both a two word op code is generated with the offset in word two. If the field with mode 6 or 7 addressing uses PC as the index register then the offset is relative to the op code + 4. For SM6 - SM7 and DM6 - DM7 combined addressing a 3 word op code is generated. Word two contains the source offset, and word 3 contains the destination offset. For SM6 = SM7 with PC the offset is relative to the op code + 4. For DM6 - DM7 with PC the offset is relative to the op code + 6.

Any autoincrements/decrements in the source are fully completed before any destination decoding begins.

#### FORMAT 11 OP CODES

All are one word op codes with a 4 bit source and a 4 bit destination argument. Each argument consists of a 3 bit register argument preceded by a 1 bit indirect argument.

## APPENDIX C

### PROGRAMMING NOTES

Several of the op codes and addressing modes have personality peculiarities that the user should be aware of. Most of these can be put to good use in particular situations. This appendix attempts to list most of them.

IEN: This instruction allows one more instruction to begin execution before enabling I2.

IDS: This instruction allows one more instruction to begin execution before disabling I2. IDS is therefore interruptable. If such a situation occurs the status of I2 that is included in the pushed PC will equal 0.

HALT: There is no halt in the microcode. A selection of options is therefore given that allows the user to define HALT for himself.

### ADDRESSING MODES

In order to clarify the function of the various addressing modes several programming examples are given. In each case assume that the first word of the op code is at location X.

#### SET R0

Register R0 is set to all ones.

#### CLR @R2

The memory location pointed to by R2 is cleared to zeros. If R2 contained a "0100" the memory word address "0100" would be cleared.

#### INC (R3)+

The memory location pointed to by R3 is incremented by one. R3 is then incremented by 2.

#### DEC (PC)+

Location X + 2 is decremented by one, and program control is advanced to location X + 4. This allows for in-line literals in a program, a method that saves a word of memory in most cases.

#### SWAB @(R4)+

If R4 contains a "0100" and location "0100" contains a "0200" then the two bytes in location "0200" are swapped and R4 is incremented to "0102".

#### COM -(R5)

R5 is decremented by two. The address specified by the altered R5 is one's complemented.

#### NEG -(PC)

A BOZO no-no since location X is the location negated and program control is again transferred to location X after the negation is completed.

#### TST @-(R1)

If R = "0104" and location "0102" contains a "1000" then the following sequence occurs: (1) R1 is decremented by 2 to "0102". (2) The contents of location "0102" (i.e. "1000") becomes the address of the operand to be tested.

#### ROR 4(R4)

The contents of memory location R4 + 4 is rotated right. R4 is not altered. Word two of this op code contains a 4. Program control is advanced to location X + 4 at the completion of the rotate.

#### ROL @6(SP)

The contents of memory location SP + 6 contains the address of the operand to be rotated. Word two of this op code contains a 6. Program control is advanced to location X + 4 at the completion of the rotate.

#### JSR PC, TAG

Location X + 2 contains the byte offset from location "TAG" to location X + 4. The address of location X + 4 is pushed onto the stack, and the address of location "TAG" is placed in PC.

#### JSR R5, TAG

Location X + 2 contains the byte offset from location "TAG" to location X + 4. The content of register R5 is pushed onto the stack, the address of location X + 4 is placed in R5, and the address of location "TAG" is placed in PC.

#### JSR PC, (R4)+

Location X + 2 is pushed onto the stack, R4 is moved to PC, and R4 is incremented by two.

#### JSR PC, @(SP)+

This is a co-routine call. Pay attention:

- 1) The contents of the location pointed to by SP is saved in CPU register "TMPA".

- 2) SP is incremented by two.
- 3) The address of location X + 2 is pushed onto the stack
- 4) CPU register "TMPA" is moved to PC

The effect of all this is to swap the top word on the stack with the address of location X + 2 without altering SP or stack size.

Consider the following routine.

```
SUBR: JSR    PC,2(PC)
TAGA: JSR    PC,@(PC)
TAGB:      .
        .
        .
      RTN    PC
```

The first JSR places the address of TAGA on the stack and executes the routine starting at TAGB. The RTN PC transfers control to location TAGA when it is executed. The second JSR places address TAGB onto the stack and into PC, effectively leaving PC unaltered. The second time the RTN PC is executed program control passes to location TAGB. The third time the RTN PC is executed program control passes back to the routine that call subroutine SUBR. Since TAGA and TAGB are never addressed explicitly both of the labels could be eliminated from the program. If left in then the "2(PC)" could be replaced with "TAGB".

CMP (R0)+, (R0)+

If R0 = "0100" then the contents of location "0100" is compared to the contents of location "0102", and R0 is incremented to "0104". All source auto increments or decrements are completed before destination decoding begins.

MOV @R2,-(R2)

If R2 = "0106" then the contents of location "0106" is moved to location "0104", and R2 is decremented to "0104".

BIT #2,@#4

The contents of absolute memory location 4 is tested against the literal value 2. This is a three word op code with word two containing a 2 and word three containing a 4. This op code works on location 4 from anywhere in memory.

CMP (PC)+,TAG

This won't work. The assembler generates a two word op code for this with the destination offset in word two. The execution of the op code, however, uses word two as a literal and word three (which does not exist) as the destination offset. By swapping the source and destinations around then an in-line literal could be used for word three, and word two would contain a valid source offset.

#### JSR PC, (PC)+

The address of location X + 4 is pushed onto the stack, and PC gets the address of location X + 2.

#### JSR R5, (PC)+

The contents of R5 are pushed onto the stack, R5 gets the address of location X + 4, and PC gets the address of location X + 2.

#### MOVB (R0)+, (R0)+

If R0 = "0102" then the contents of memory byte location "0102" is moved to memory byte location "0103", and R0 is incremented to "0104".

#### MOVB (SP)+, R1

The contents of the memory byte addressed by SP is moved to the lower byte of R1, the sign bit (bit 7) is replicated through bit 15 of R1, and SP is incremented by 2. SP is always autoincremented or autodecremented by two.

#### CLRB (PC)+

The contents of the lower byte memory location X + 2 is cleared to zeros. The upper byte (X + 3) is not affected. PC is incremented by two. PC is always autoincremented or autodecremented by two.

#### BISB R0, R1

The lower bytes of register R0 is logically ORED with the lower byte of register R1. The upper byte of R1 is not altered.

#### MOVB @(R2)+, @-(R3)

If R2 contains a "0100" and R3 contains a "0200" then location "0100" contains the byte address of the source operand and location "01FE" contains the address of the destination byte that is to receive the source byte. R2 is incremented by two, and R3 is decremented by two since they point to addresses of (16 bit) addresses.

#### JSR SP, TAG

Not recommended since the value of the stack is lost. Perfectly legal however.

### SAVS and RSTS

Although designed to be used for automatic register and I/O priority level saving and restoring, the lack of hardware priority masking does not alter the operation or the op codes. The SAVS op code is usually the first instruction executed in a device interrupt routine, and the RSTS is the last. The priority mask can use a one bit as an enable or disable with bit 0 the highest or lowest priority level. Such decisions are made by the hardware.

## POWER FAIL

Two levels of power fail are provided for in the firmware. The hardware may use two, one, or no levels of power fail. The three modes are discussed in increasing order of complexity.

NO LEVELS: External address register bit 7 is hardwired to 0, and a prayer is offered.

ONE LEVEL: The detection of a power fail sets bit 7 of the external status register and the CPU RESET line. When the power fail disappears the CPU RESET line is reset, but bit 7 of the external status register remains set. The Line Clock Clear State Code (see appendix D) clears bit 7 of the external status register (and bits 5, 6 if used). A system power up is then executed.

TWO LEVELS: This requires two hardware functions, AC LOW and DC LOW, plus two levels of power fail; AC and DC. It all works like this: If AC power begins to deteriorate AC LOW is set first. This sets bit 7 of the external status register and generates an interrupt via I0 or I1. If AC power does not deteriorate too far then nothing else happens except that bit 7 of the external status register is reset when power is restored. If AC power continues to deteriorate then eventually DC power will begin to deteriorate. When this happens DC LOW is set and DC LOW sets CPU RESET. AC LOW is still set and it maintains bit 7 of the external status register. When power is restored DC LOW is reset. This resets CPU RESET. A power up sequence is initiated, and the Line Clock Clear State (see appendix D) clears The External Status Register bit 7 (plus 5 and 6 if they are used). If the user wishes to be able to execute a programmed power fail routine even during a sudden and complete power failure then the DC power supply must be strong enough to run the CPU and MEMORY for at least 2 milliseconds. The power fail interrupt must also be programmed, and the interrupts enabled.

The use of the Line Clock Clear State Code to clear bits 5-7 on a CPU RESET function (plus the line clock of course) should have no effect on normal system operation. Should an error occur during a non-vectored interrupt the error would be cleared momentarily and then set again as CPU RESET obviously could not have been generated. If it had been then the system could not be in the non-vectored interrupt routine.

## PARITY AND BUS ERRORS

These functions are also part of the CPU RESET function along with power fail/up. In order to get only one or the other then bit 7 of the external status register must be reset when the CPU RESET function

is activated. In order to generate a valid CPU RESET the CPU RESET line must be held active for three clock cycles. Longer is fine, but the CPU goes into a wait state until the CPU RESET is reset. If more than one error exists at one time then the highest priority error is the one honored. The priority, from highest to lowest, is:

Power Fail  
Bus Error  
Parity Error

If all 3 functions are reset a power up is assumed. All 3 functions have a bit associated with them in the external status register. Once set these bits stay set until cleared by the Line Clock Clear State Code (see appendix D) that is generated during the first phases of the reset routine. See chapter two "Power Up Options".

## APPENDIX D

### MICROM STATE CODE FUNCTIONS

Below is a list of MICROM STATE CODE FUNCTIONS for the WD1600 with a brief description of what each does. More elaborate descriptions, where necessary, follow the table.

<u>CODE</u>	<u>MNEMONIC</u>	<u>FUNCTION</u>
0001	PMSK	Priority mask out
0010	RUN	Macro instruction fetch
0011	IORST	I/O reset
0100	INTEN	I2 set
0101	INTDS	I2 reset
0110	ESRR	External status register request
0111	SRS	System reset
1000	BYTE	Read byte operation
1001	RMWW	Read-modify-write word
1010	RMWB	Read-modify-write byte
1011	RLCI	Reset line clock interrupt
1100	EARR	External address register request
1101		Duplicate of "BYTE"
1110		Duplicate of "RMWW"
1111		Duplicate of "RMWB"

PMSK: The state code is generated on an OUTPUT WORD instruction when a new mask is written into location "2E". It signals the I/O devices that a new interrupt mask is on the DAL.

RUN: Generated during macro instruction fetch for a run light.

IORST: Generated during a RESET macro op code to reset I/O devices to some preset state.

INTEN: Enables the interrupt enable line -I2.

INTDS: Disables the interrupt enable line -I2.

ESRR: Generated during an INPUT STATUS BYTE micro op code to indicate that the external status register is being requested. See note 1.

SRS: Generated during a power up for a master system reset. This code is followed by a 300 cycle wait to allow time for any reset functions the hardware generates to be completed before any DAL requests are generated.

BYTE: Generated during an INPUT BYTE micro op code to indicate a read byte operation without a read-modify-write.

RMWW: Generated during an INPUT WORD micro op code with RMW active to indicate a read-modify-write word sequence.

RMWB: Generated during an INPUT BYTE micro op code with RMW active to indicate a read-modify-write byte sequence.



RLCI: Generated during a CPU RESET or a non-vectorized interrupt without a power fail to clear both the line clock interrupt and external status register bits 5-7.

EARR: Generated during an INPUT STATUS BYTE micro op code to indicate a request for the external address register during the user bootstrap routine.

CODES "D" - "F": Duplicates of codes "8" - "A" respectively except that these codes appear as a part of the READ micro op codes instead of as a part of the INPUT micro op codes. Either or both may be used by the hardware as is convenient. These codes precede the others. They are generated only once, however, instead of repeating in the event of a wait state as the others do.

NOTE 1: INPUT STATUS BYTE is not a function of reply and does not generate a SYNC. For these reasons the DAL must be tri-stated if a DMA device also exists. The data is always gated onto the lower byte. The upper byte is ignored.

NOTE 2: Lack of state codes "8" - "A" or "D" - "F" during a READ - INPUT sequence implies a read word operation without read-modify-write.

## APPENDIX E

### OP CODE TIMINGS

All times are in cycles. Timings include all OP Code fetches, memory reads, and memory writes applicable to each. Timings assume that the memory is running with full speed with respect to the CPU. This requires a 16 Bit access time = 1 CPU cycle, and a 16 Bit memory read/write cycle time = 2 CPU cycles. One CPU cycle = 300 NS @ 3.3 MHZ, 400 NS @ 2.5 MHZ, and 500 NS @ 2 MHZ clock rates. Timings are included for SM0 and DM0 as basic with additions as necessary in tables that follow the OP Codes for SM1-7 and DM1-7 timings.

#### FORMAT ONE OP CODES

<u>OP CODE</u>	<u># CYCLES</u>
NOP	10
RESET	10
IEN	10
IDS	10
HALT	16+
XCT	44 + OP CODE EXECUTED
BPT	24
WFI	16+
RSVC	62
RRTT	60
SAVE	46
SAVS	65
REST	48
RRTN	52
RSTS	64
RTT	13

#### FORMAT TWO-FOUR OP CODES

<u>OP CODE</u>	<u># CYCLES</u>
IAK	10
RTN	12
MSKO	10
PRTN	22
LCC	7
SVCA	37
SVCB	73
SVCC	71

#### FORMAT FIVE OP CODES

All branches = 9 cycles if branch occurs or not.

# FORMAT SIX OP CODES

<u>OP CODE</u>	<u># CYCLES</u>
ADDI	9
SUBI	9
BICI	9
MOVI	9
SSRR	8 + (5 X # bits shifted)
SSLR	8 + (5 X # bits shifted)
SSRA	8 + (7 X # bits shifted)
SSLA	8 + (5 X # bits shifted)
SDRR	20 + (7 X # bits shifted)
SDLR	20 + (7 X # bits shifted)
SDRA	20 + (9 X # bits shifted)
SDLA	20 + (7 X 3 bits shifted)

# FORMAT 7 OP CODES - DMØ

<u>OP CODES</u>	<u># CYCLES</u>	<u>OP CODES</u>	<u># CYCLES</u>
ROR	1Ø	RORB	9
ROL	1Ø	ROLB	9
TST	1Ø	TSTB	9
ASL	1Ø	ASLB	9
SET	1Ø	SETB	1Ø
CLR	1Ø	CLRB	9
ASR	12	ASRB	11
SWAB	1Ø	SWAD	21
COM	1Ø	COMB	9
NEG	1Ø	NEGB	9
INC	1Ø	INCB	9
DEC	1Ø	DECB	9
IW2	1Ø	LSTS	15
SXT	12	SSTS	1Ø
TCALL	21	ADC	11
TJMP	16	SBC	11

## FOR WORD OPS AND:

DM1	ADD	4
DM2	ADD	4
DM3	ADD	8
DM4	ADD	6
DM5	ADD	1Ø
DM6	ADD	1Ø
DM7	ADD	14

## FOR BYTE OPS AND:

DM1	ADD	3
DM2	ADD	3 *
DM3	ADD	7
DM4	ADD	5 *
DM5	ADD	9
DM6	ADD	9
DM7	ADD	13

For DM1 - DM7 and:

CLR subtract 1 cycle  
SWAB subtract 1 cycle

\*NOTE: Add 2 more if SP or PC.

### FORMAT 8 OP CODES

<u>OP CODE</u>	<u># CYCLES</u>	<u>(ASSUMES NO INTERRUPTS)</u>
MBWU	17 + (16 X # words moved)	
MBWD	15 + (16 X # words moved)	
MBBU	17 + (15 X # bytes moved)	
MBBD	15 + (15 X # bytes moved)	
MBWA	19 + (16 X # words moved)	
MBBA	19 + (15 X # bytes moved)	
MABW	19 + (16 X # words moved)	
MABB	19 + (15 X # bytes moved)	

### FORMAT 9 OP CODES - DMØ

<u>OP CODE</u>	<u># CYCLES</u>
JSR*	22
LEA*	15
ASH	19 if DST = Ø; 22 + (5 X count) if DST>Ø; 25+(7 X count) if DST < Ø.
SOB	1Ø if no branch, 13 if branch
XCH	23
ASHC	19 if DST = Ø; 38 + (7 X count) if DST>Ø; 38+(9 X count) if DST < Ø
MUL	183
DIV	29 if divisor error, 2Ø2 if no divisor error

\*NOTE: DMØ illegal. Used as base figure only.

FOR ALL OP CODES EXCEPT SOB AND:

DM1 add Ø  
DM2 add 2  
DM3 add 2  
DM4 add 2  
DM5 add 4  
DM6 add 4  
DM7 add 8

### FORMAT 1Ø OP CODES - SMØ AND DMØ

<u>OP CODE</u>	<u># CYCLES</u>
ADD	11
SUB	11
AND	11
BIC	11
BIS	11
XOR	11
CMP	11
BIT	11
MOV	11
CMPB	11
MOVB	12
BISB	11

For SM1: add 3 for word ops, 1 for byte ops.  
 For SM2: add 4 for word ops, 2 for byte ops. \*  
 For SM3; add 7 for word ops, 5 for byte ops.  
 For SM4; add 5 for word ops, 3 for byte ops. \*  
 For SM5; add 9 for word ops, 7 for byte ops.  
 For SM6; add 9 for word ops, 7 for byte ops.  
 For SM7; add 13 for word ops, 11 for byte ops.

For DM1; add 4 for word ops, 3 for byte ops.  
 For DM2; add 4 for word ops, 3 for byte ops. \*  
 For DM3; add 8 for word ops, 7 for byte ops.  
 For DM4; add 6 for word ops, 5 for byte ops. \*  
 For DM5; add 10 for word ops, 9 for byte ops.  
 For DM6; add 10 for word ops, 9 for byte ops.  
 For DM7; add 14 for word ops, 13 for byte ops.

For MOVB and DM1-DM7 subtract 1 cycle.

\*NOTE: Add 2 if SP or PC

#### FORMAT 11 OP CODES - ALL ADDRESSING MODES

<u>FADD</u> :	If exponent difference > 39	:	138-145
	Worst Case	:	638
	Typical	:	180-420
<u>FSUB</u> :	If exponent difference > 39	:	141-148
	Worst Case	:	641
	Typical	:	190-430
<u>FMUL</u> :	If either operand = 0	:	108-111
	Worst Case	:	805
	Typical	:	590-780
<u>FDIV</u> :	If divide by 0	:	96
	If divide into 0	:	118
	Worst Case	:	1596
	Typical	:	280-1210
<u>FCMP</u> :		:	49-86